Checkpointing and Rollback-Recovery of Distributed Applications in Kubernetes

Europe Research Interest Group Meeting - April 13, 2023

Radostin Stoyanov
DPhil Candidate, Scientific Computing Group
Supervisor: Prof. Wes Armour
Square Kilometer Array

SKA1-Low (Australia)  SKA1-Mid (South Africa)  a) LOFAR-INTL Snapshot  b) SKA-Low Snapshot

1 LOFAR (LOw Frequency ARray) - a radio telescope working at the lowest frequencies accessible from Earth.
Five-Year Failure Record of a Leadership-Class Supercomputer

Titan supercomputer, a Cray XK7 system located at the Oak Ridge Leadership Computing Facility (OLCF)

- 18,688 nodes (total of 299,008 CPU cores)
- Combined elements provide 17.59 petaFLOPs
- Each node has:
  - 32 GB of main memory
  - AMD 16-core Opteron CPU
  - NVIDIA Tesla K20 GPU

Categorization of failures over a 5 year (2014 - 2018) reliability record:

1. GPU-related errors dominate the record
2. Most failures only affect a single node
3. Software failures bring down several nodes
Checkpoint/Restore in Userspace

- Open-source tool for checkpoint/restore of Linux processes
- Entirely in userspace
- Integrated with:
  - runc / crun / youki
  - Docker / Podman / LXD
  - CRI-O / containerd
  - Kubernetes
Container Checkpointing in Kubernetes

An alpha feature introduced with Kubernetes v1.25
https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha
Container Checkpointing in Kubernetes

Deployment may consist of multiple Pods
Container Checkpointing in Kubernetes
Container Checkpointing in Kubernetes

Container dependencies must be started before the container is started.
Container Checkpointing in Kubernetes

Kubernetes Controller
- kube-apiserver
- kube-scheduler
- kube-controller-manager
- etcd

Node
- kubelet
- CRI-O
- runc

Pod
- Infrastructure Container
- Container A
- Container B
- Container C

Checkpoint
Container Checkpointing in Kubernetes

Node
- kubelet
- CRI-O
- runc
- Pod
  - Container

Pod
- Infrastructure Container
  - Container A
  - Container B
  - Container C

Checkpointing:
1. **Checkpoint**
2. **Restore**

Diagram:
- Kubernetes Controller
  - kube-api-server
  - kube-scheduler
  - kube-controller-manager
  - etcd

- Container dependencies
  - Container A
  - Container B
  - Container C
Container Checkpointing in Kubernetes

Kubernetes Controller
- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager

Node
- kubelet
- CRI-O
- runc

Pod
- Container
- Infrastructure Container

Container Engine
- Container Runtime

Container Runtime
- Container
Container Checkpointing in Kubernetes
Container Checkpointing in Kubernetes

Kubernetes Controller
- etcd
- kube-scheduler
- kube-controller-manager
- kube-apiserver

Node
- kubelet
- CRI-O
- CRIU
- runc

Container Registry
- Pod
- Pod
- Pod

Container Engine
- Container Runtime

Checkpoint
- descriptors.json
- config.json
- devshm-checkpoint.tar
- reg-files.img
- core-%u.img
- cores-%u.img
- pages.img
- inotify.img
- pipes.img
- pipes-data.img
- tmpfs-dev-%u.tar.gz.img
- ...

Buildah
- (commit)
- (push)

Buildah
- OCI Checkpoint Image

Sensitive Data
Checkpoint Images

Basic units of deployment using standard image distribution infrastructure.
Checkpoint Images

Dec, 2022. Adrian Reber. Forensic container checkpointing in Kubernetes. [https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/](https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/)
Checkpont Images

$ podman container checkpoint --create-image <img>

Dec, 2022. Adrian Reber. Forensic container checkpointing in Kubernetes. [https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/](https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/)
$ podman container checkpoint --create-image <img>

Dec, 2022. Adrian Reber. Forensic container checkpointing in Kubernetes. [https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/](https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/)
Checkpoint Images

$ podman container checkpoint --create-image <img>

Dec, 2022. Adrian Reber. Forensic container checkpointing in Kubernetes. [https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/](https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/)
$ podman container checkpoint --create-image <img>

1. Checkpoint
   - crun
   - libcriu
   - crio

2. Build & Commit
   - buildah
   - OCI image

Node
- kubelet
- CRI-O
- runc

Pod
- C1
- C2
- C3

Registry
- Checkpoint
- Config, Logs, Metadata
- CRIU Images

Dec, 2022. Adrian Reber. Forensic container checkpointing in Kubernetes. [https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/](https://kubernetes.io/blog/2022/12/05/forensic-container-checkpointing-alpha/)
Coordinated Checkpointing

Maintaining the consistency of distributed computations in the presence of failures.
Coordinated Checkpointing Model

- **Two-phase** algorithm based on Koo–Toueg\(^1\)
- Checkpoints: **tentative** (local) and **permanent** (global)
- In case of a failure, rollback to the last permanent checkpoint

---

Coordinated Checkpointing Model

Phase 1

1: Create a tentative checkpoint for each container.

2: Notify the coordinator whether it succeeded.

3: If all tentative checkpoints are successful, continue to Phase 2. Otherwise, discard tentative checkpoints and go back to Step 1.
Coordinated Checkpointing Model

Phase 2

1: All tentative checkpoints are made permanent.
2: Transfer local checkpoints to coordinator.
3: Create OCI checkpoint images.
4: Upload images to registry.
CRIU Coordinator

- Per container CRIU configuration file \(^1\)
- An external executable running as an action script \(^2\)
- Checkpoint synchronization at pre-dump or pre-stream \(^3\)
- Environment variables config \(^4\):
  - Container/Pod ID
  - Dependencies
  - Address & Port
  - Log

---

\(^1\) [https://github.com/opencontainers/runc/blob/main/docs/checkpoint-restore.md](https://github.com/opencontainers/runc/blob/main/docs/checkpoint-restore.md)

\(^2\) [https://criu.org/Action_scripts](https://criu.org/Action_scripts)

\(^3\) action-scripts: add pre-stream hook ([https://github.com/checkpoint-restore/criu/pull/2149](https://github.com/checkpoint-restore/criu/pull/2149))

\(^4\) config: add --action-env cli option ([https://github.com/checkpoint-restore/criu/pull/2136](https://github.com/checkpoint-restore/criu/pull/2136))
Checkpoint Streaming

Nicolas Viennot (Two Sigma). Fast checkpointing with criu-image-streamer. Linux Plumbers Conference. 2020
Radostin Stoyanov, et. al., Secure Image-less Container Migration. Linux Plumbers Conference. 2019
Rodrigo Bruno, et. al., GC-assisted JVM Live Migration for Java Server Applications. Middleware '16
Coordinated Checkpointing Model
Global Checkpoint

Ensuring consistency and correctness of distributed applications.
Preemptive Scheduling

Global Checkpoint

Control Plane
- scheduler
- etcd
- api-server
- controller-manager

Pod
- kubelet
- Pod

Scheduling Cycle
- Sort
- Pre Filter
- Filter
- Score
- Normalize Score
- Reserve
- Permit
- Wait on Permit
- PreBuild
- Build
- PostBuild

Binding Cycle

https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/
Preemptive Scheduling

Application Fault Tolerance
Application Fault Tolerance

root@skatelescope-flat-sdp-console-0:/app# ska-sdp --help
Command line utility for interacting with SKA Science Data Processor (SDP).

Usage:
ska-sdp COMMAND [options] [SDP_OBJECT] [<args>...]
ska-sdp COMMAND (-h|--help)
ska-sdp (-h|--help)

SDP Objects:
  pb          Interact with processing blocks
  script      Interact with available processing script definitions
  deployment  Interact with deployments
  eb          Interact with execution blocks
  controller  Interact with Tango controller device
  subarray    Interact with Tango subarray device

Commands:
  checkpoint  A fault-tolerance mechanism periodically saving the runtime state of execution block
  list        List information of object from the Configuration DB
  get         Print all the information (i.e. value) of a key in the Config DB
  create      Create a new, raw key-value pair in the Config DB;
               Run a processing script; Create a deployment
  update      Update a raw key value from CLI
  edit        Edit a raw key value from text editor
  delete      Delete a single key or all keys within a path from the Config DB
  end         Stop/Cancel execution block
  import      Import processing script definitions from file or URL

https://gitlab.com/ska-telescope/sdp/ska-sdp-config.git
Application Fault Tolerance

root@ska-sdp-console-0:/app# ska-sdp checkpoint --help
A fault-tolerance mechanism that periodically saves the runtime state of processing block(s) to persistent storage. In the case of failure, the processing block(s) will be restarted from the most recent checkpoint and resume processing.

Usage:
ska-sdp checkpoint (-a |--all) list
ska-sdp checkpoint [options] interval [<interval>]
ska-sdp checkpoint [options] storage [<location>]
ska-sdp checkpoint (-h|--help)

Arguments:
<interval> Checkpoint interval in milliseconds.
<location> The location where checkpoint snapshots are made durable.

https://gitlab.com/ska-telescope/sdp/ska-sdp-config.git
Rollback Recovery

Recovering from failures and continue to operate correctly.
Rollback Recovery

$ sudo cat /etc/crJu/runc.conf

tls
tls-cacert=/etc/pki/CA/cacert.pem
tls-cert=/etc/pki/pNu/crJu/servercert.pem
tls-key=/etc/pki/pNu/crJu/private/serverkey.pem

$ crJu dump --tls -t <PID> -D /var/lib/containers/storage/<container ID>

$ crJu restore --tls -D /var/lib/containers/storage/<container ID>
Rollback Recovery

**Domino effect & Livolock problem**

- Process $P_1$
- Process $P_2$
- Process $P_3$

- $C_1$, $C_2$, $C_3$
- $m_1$, $m_2$, $m_3$, $m_4$
- Failure
- Rollback

**Consistent state**
- Message in-transit
- Transmitted message
- Transient local checkpoint
- Permanent local checkpoint
- Global checkpoint

**Inconsistent state**
- $P_1$, $P_2$, $P_3$
Rollback Recovery

Phase 1:

1: Create containers from checkpoint images.
2: Restore synchronization at pre-restore hook.
3: Notify the coordinator whether it succeeded.
4: If all containers are successfully restored, continue to Phase 2. Otherwise, remove containers and go back to Step 1.
Rollback Recovery

Phase 2:
1: Restart synchronization at pre-resume hook.
2: Notify the coordinator with result.
Rollback Recovery

Correctness

• Either *all* or *none* of the containers have permanent checkpoints.
• All containers resume execution from a consistent state.
• During recovery, execution is resumed only when all containers have been successfully restored.
Summary & Questions

https://criu.org/
https://github.com/checkpoint-restore/criu