Checkpoint/Restore SIG Node Update

Adrian Reber Mrunal Patel

2021, March 23

Checkpointing - Pod Lifecycle - 1

- Checkpointing possible at any time
  - After init container has finished
- Checkpointing is triggered by the container engine
  - CRI-O in our case
- Container runtime (`runc`, `crun`) triggers CRIU
- CRIU writes the checkpoint to a given directory
- Container engine creates checkpoint archive
  - The CRIU checkpoint of all container processes
  - The file system diff between the OCI image and now

Checkpointing - Pod Lifecycle - 2

- Pod checkpoint archive contains
  - Checkpoint archive for each container
  - Pod Metadata
- A checkpointed container can be restored
  - Multiple times
  - In different Pods
- Pod checkpoint archive is created in `/var/lib/kubelet/checkpoints`
- Kubelet adds additional metadata
- Checkpointing currently only possible via `kubectl`

Checkpoint Size

- Used memory by all processes in the container
- and some kilobytes for each process
- All files that changed (compared to the initial OCI images)
- Everything is part of one single tar archive

Checkpoint Storage

- Directory to store the Pod checkpoint archives
- Network mounted for a possible migration use case

Other Approaches

- Not integrated into Kubernetes
  - Privileged sidecar to the checkpointing
  - Privileged init container to restore
- Too complicated as CRIU needs access to runtime/engine internal information
- CRIU already well integrated into runtimes and engines

Hooks

- Initial discussions to inform applications about being checkpointed
- Possible use case: JVM

Use Cases - 1

- Checkpointing for maintenance
  - Install new kernel
  - Checkpoint
  - Reboot
  - Restore without losing state
- Fast startup: create copies of initialized containers
- Migration

Use Cases - 2

- JVM: fast startup of initialized JVMs
- Mathworks uses start from checkpoint to decrease startup time by 5 minutes
- Keep containers running while migrating to another storage system