Java and CRIU

Challenges and Opportunities

Christine H. Flood
chf@redhat.com

Red Hat

What is CRIU?

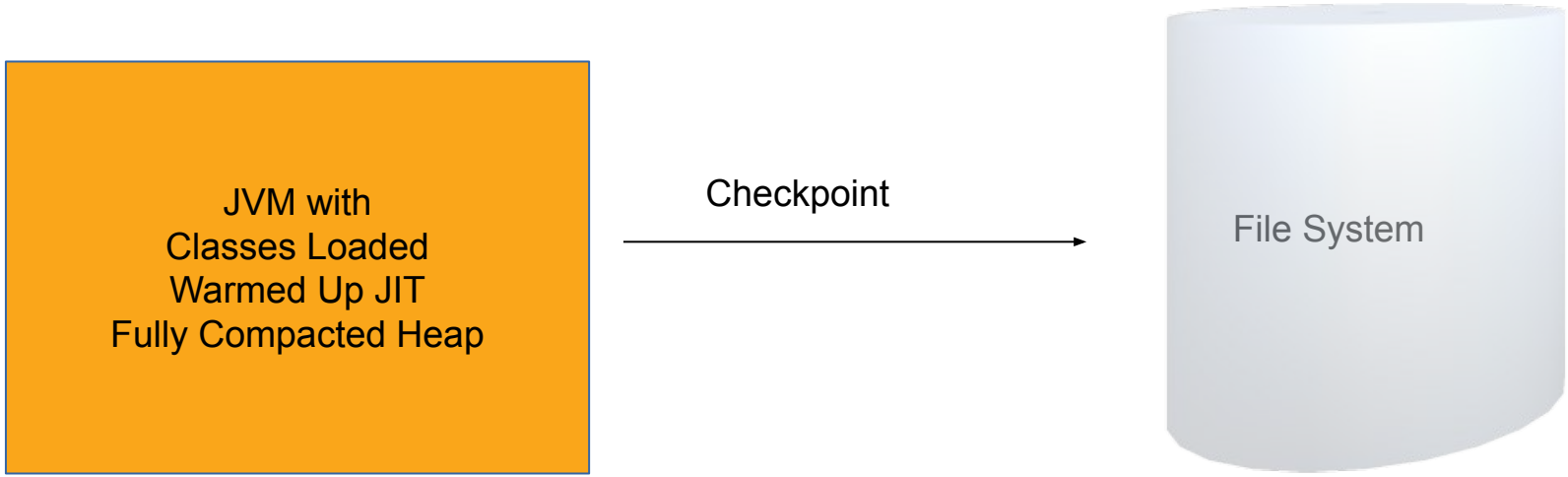
Checkpoint/Restore in Userspace

- Linux utility which copies entire process state into files.
- Files can be quickly restored on different hosts.
- Handles open files/sockets/almost all the gory details.

Find out more https://criu.org/Main_Page

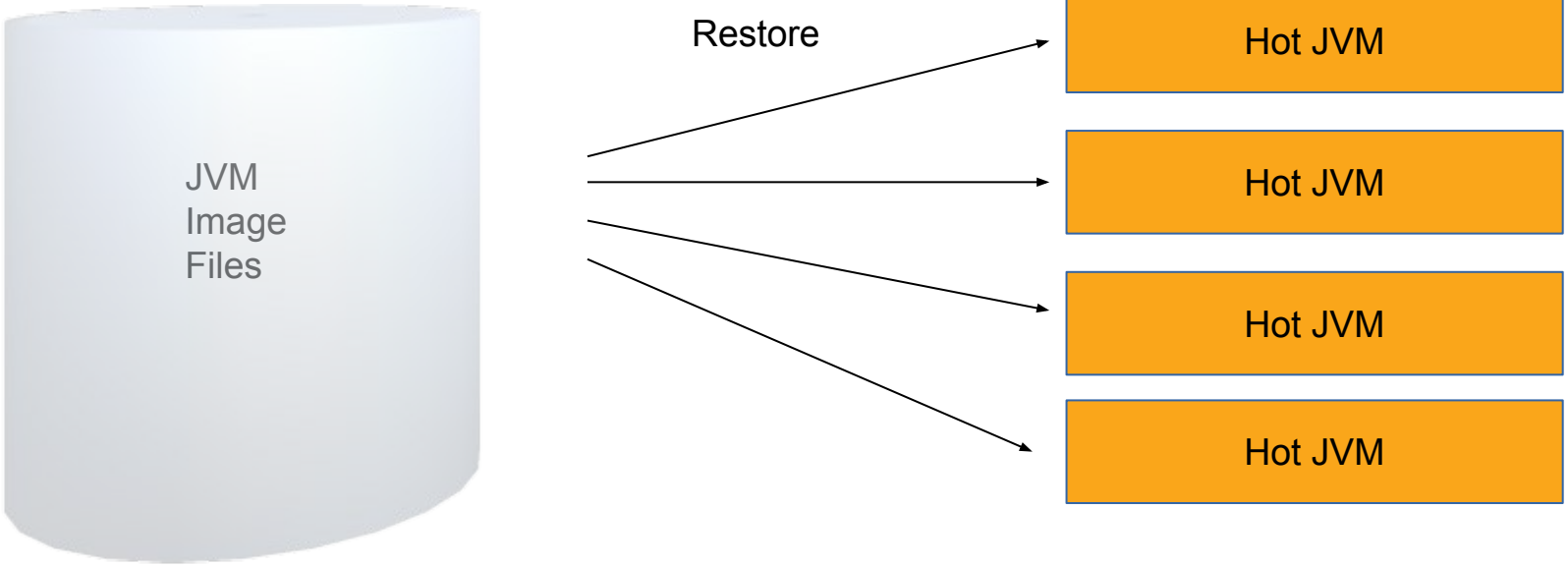
Red Hat

Motivation



Red Hat

Motivation



Red Hat

Improved startup time

/usr/bin/time java -XX:-UsePerfData -XX:+UseSerialGC TestRandomNoCheckpointing 10000 10000000000
Testing random number generator with upper bound = 10000 and sample size 10000000000
Max bin size = 1003474 Min bin size = 10000 median bin size = 1000000.0
275.29user 0.03system 4:35.83elapsed 99%CPU (0avgtext+0avgdata 36724maxresident)k0inputs+0outputs (0major+5351minor)pagefaults 0swaps

/usr/bin/time sudo java -cp ./:./checkpoint.jar -XX:+UseSerialGC -XX:-UsePerfData TestRandomRestore /home/chf/SavedWorlds/run1591741920734
RestoreTheWorld: file = /home/chf/SavedWorlds/run1591741920734 fd = 6
Testing random number generator with upper bound = 10000 and sample size 10000000000
Max bin size = 1003939 Min bin size = 10000 median bin size = 1000000.0
0.21user 0.04system 0:00.19elapsed 135%CPU (0avgtext+0avgdata 34224maxresident)k
0inputs+168outputs (0major+11568minor)pagefaults 0swaps

Red Hat

Improved Footprint

4 unique java processes

pid	virt	res	shr
17179	2797264	29604	16496
17194	2797264	30304	16888
17164	2797264	30072	16668
17209	2797264	30888	16948

4 restored instances of the same java process

pid	virt	res	shr
17650	2797260	30784	16880
17696	2797260	16656	2752
17725	2797260	16652	2752
17753	2797260	16656	2752

Contributions of This Work

Experiments with CRIU and Java so you don’t have to.

Checkpoint.jar which provides calls to CRIU from Java and enables precise checkpoints. Byteman allows precise Checkpoints in unmodified existing code base.

Hooks to add to your program to shutdown network connections and close files.

CheckpointRestore.java

Specify precisely where Checkpointing occurs.

Add hooks before checkpointing / after restoring

Simple Example

```
import org.checkpoint.*;
class Test1 {
    public static void main(String[] args) {
        String dir = "world" + System.currentTimeMillis();
        try {
        CheckpointRestore.saveTheWorld(dir);
        …
        System.out.println(“Hello World”);
}
```

Red Hat

Simple Example

javac -cp ./:./checkpoint.jar Test1.java

Setsid java -XX:-UsePerfData -XX:+UseSerialGC cp ./:checkpoint.jar Test1

Restore via command line:

cd <savedir>

sudo criu restore --shell-job -o restore.log

Restore via Java

CheckpointRestore.restoreTheWorld(<savedir>);

Checkpoint files

The java library creates status files in the saved directory:

- Save.log
  - Gives you the details of what happened during checkpointing
- Restore.log
  - Gives you details about restoring.

Adding Hooks

```
class BeforeHook extends Hook {
    public void run() {
      System.gc();
      System.gc();
      System.out.println("GCing before checkpoint");
    }
}
```

Red Hat

Adding Hooks

```
class Test2 {
    public static void main(String[] args) {
        String dir = "world" + System.currentTimeMillis();
        CheckpointRestore.registerCheckpointHook(new BeforeHook());
        try {
                CheckpointRestore.saveTheWorld(dir);
        } catch (Exception e) {
        }
        System.out.println("Hello World");
...
```

Adding Checkpoints via Byteman

Byteman is a tool for injecting Java code into methods without recompiling.

https://byteman.jboss.org/

Byteman rule file MyFile.btm

```
RULE trace mymethod entry
CLASS Shaggy
METHOD mymethod
AT ENTRY
IF true
DO traceln("mymethod"); org.checkpoint.CheckpointRestore.saveTheWorld("./tmp"  +
System.currentTimeMillis())
ENDRULE
```

Triggering checkpoints via byteman

```
setsid java -javaagent:/home/chf/byteman/byteman-download-4.0.14/lib/byteman.jar=script:MyFile.btm
-cp ./:checkpoint.jar -XX:+UseSerialGC -XX:-UsePerfData Shaggy
```

Red Hat

More complicated example (Quarkus TODO app)

```
public class Todo extends PanacheEntity {

...
      public static List<Todo> findCompleted() {
        return list("completed", true);
    }
    public static long deleteCompleted() {
        return delete("completed", true);
    }


}
```

More complicated example (Quarkus TODO app)

```
RULE trace deleteCompleted exit

CLASS io.quarkus.sample.Todo

METHOD deleteCompleted

AT ENTRY

IF true

DO traceln("mymethod");  org.checkpoint.CheckpointRestore.saveTheWorld("./tmp"  + System.currentTimeMillis())

ENDRULE
```

More complicated example (Quarkus TODO app)

Checkpoint.jar added to local maven repository and dependencies added to todo-app.

```
java
-javaagent:/home/chf/byteman/byteman-download-4.0.14/lib/byteman.jar=script:TODOCheckpoint.btm
-XX:-UsePerfData -XX:+UseSerialGC -XX:TieredStopAtLevel=1 -Xverify:none -Xdebug
-Xrunjdwp:transport=dt_socket,address=0.0.0.0:5005,server=y,suspend=n
-Djava.util.logging.manager=org.jboss.logmanager.LogManager -jar
/home/chf/Nov2020/take2/quarkus-todo-app/target/todo-backend-1.0-SNAPSHOT-runner.jar
```

Quarkus TODO app

Everytime the deleteCompleted method is called, a checkpoint is created.

Additional Use Cases for Precise checkpointing

Recurring memory corruption bug.

Recurring operational bug that takes days to trigger.

Replayable log of program activity

Repository

https://github.com/chflood/JavaCriuJar

How it works

Build using Maven on your architecture.

Compiles C libcheckpointrestore.so library.

Builds a checkpoint.jar that loads libcheckpointrestore.so

Presents a Java interface to the user.

Java Challenges

JNI Library is architecture specific and violates Write Once Run Anywhere.

Current Status

Project proposal in place for Linux distribution.

But what about?

Root Access

There are patches working their way upstream to fix this.

PIDs and multiple restore:

As Root:

```
# unshare -p -m -f bash

# mount -t proc none /proc/

# criu restore -d -vvv -o restore.log && echo ok
```

Address Space Layout Randomization

Red Hat

Difference with CRAC

CRAC

JVM must be prepared to checkpoint at any time.

Java CRIU

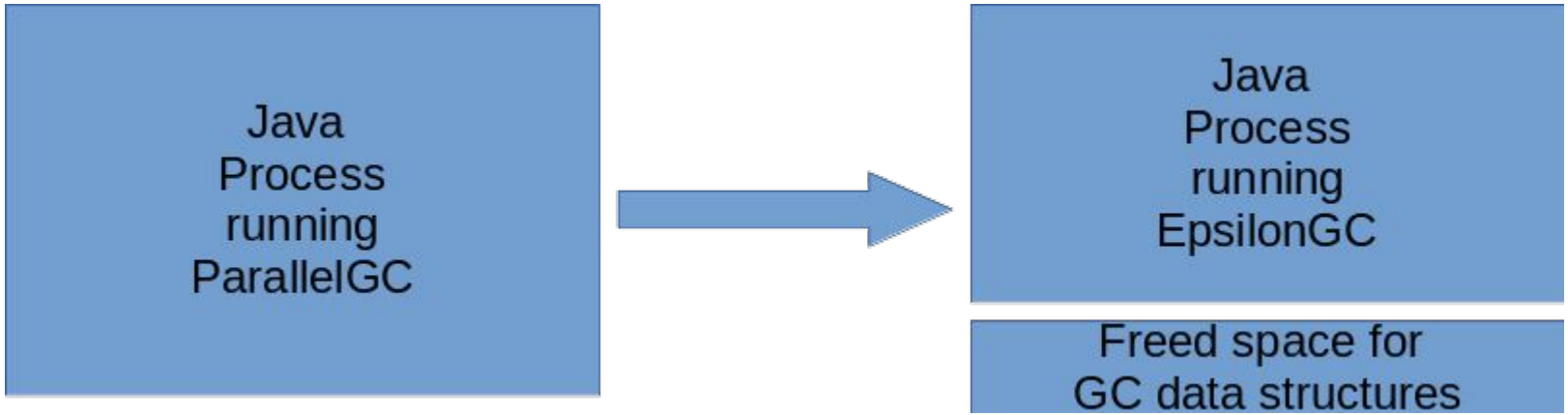
User specifies where checkpoints occur.

Future Work Containers

We are working on restoring checkpointed Java processes inside of a podman container.

Future Work

Hot Swap GCs



Red Hat

Future Work

Repair incorrect architecture assumptions.

Checkpoint on a machine with 8 processors.

Restore on a machine with 2 processors.

Fix machine characteristics so things like workstealing threads are optimal.

Future Work

Optimize Heap Layout

Inspired by Remix: Online Detection and Repair of Cache Contention For the JVM -Eizenberg, Hu, Pokam, and Devietti.

Monitor Hardware performance counters to detect things like false cache line sharing and then pad the data layout.

Contact Info

CHF@redhat.com

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make
Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat