Improving JVM Application Migration and Profiling with Checkpoint/Restore

Rodrigo Bruno, Paulo Ferreira

rbruno@gsd.inesc-id.pt, paulo.ferreira@inesc-id.pt

INESC-ID - Instituto Superior Técnico, ULisboa

Checkpoint/Restore - LinuxPlumbers'17 - Los Angeles

About Me

PhD student at University of Lisbon

- Supervised by Prof. Paulo Ferreira

Researcher at INESC-ID

Work on improving JVM ability to adapt to Big Data workloads

- JVM live migration
- Reduce JVM application latency due to Garbage Collection

Contributed to CRIU with remote images

Collaborations/Interships: Feedzai, Jelastic

Now at Microsoft Research (internship) hacking the nouveau dev. driver

This talk

Research use-cases for Checkpoint-Restore

- Runtime aware live migration
  - ALMA (published on Middleware'16, Trento)

- Efficient Application Objects Lifetime Profiler
  - POLM2 (to appear on Middleware'17, Las Vegas)

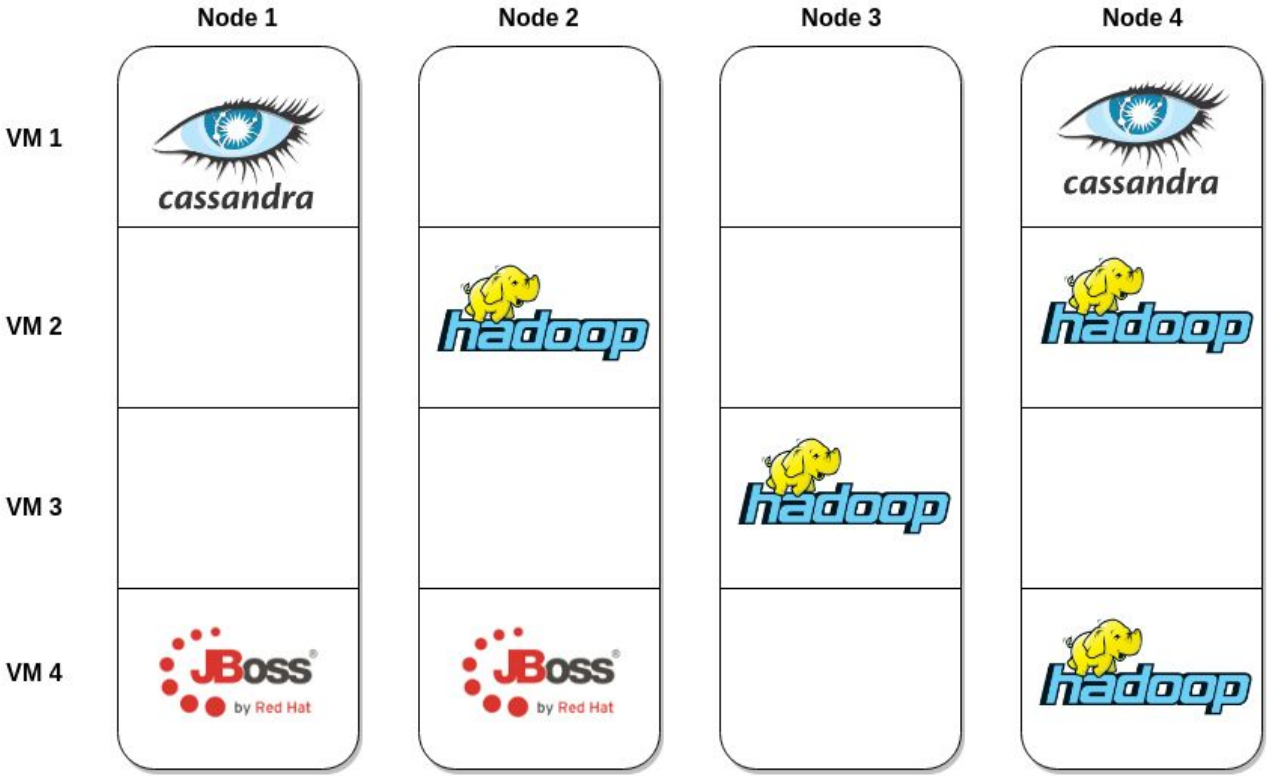
TÉCNICO LISBOA

gsd inesc id lisboa

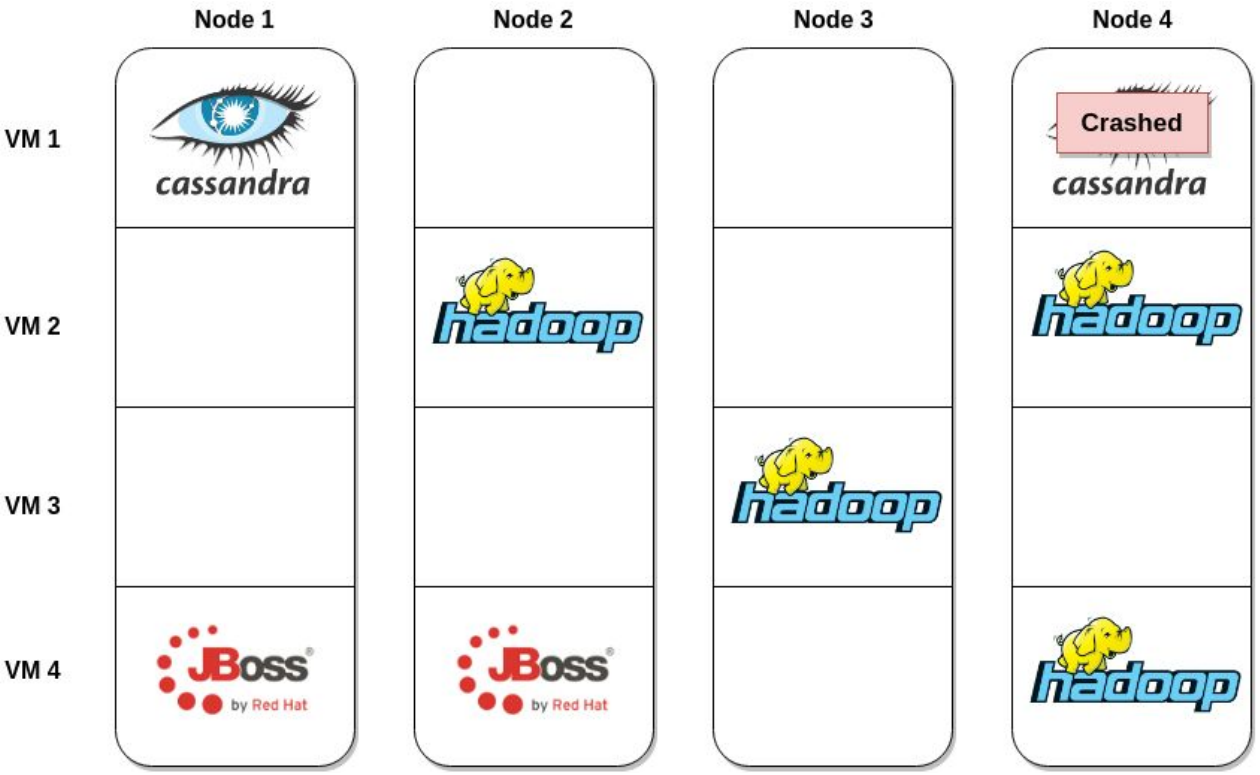
ALMA - JVM Live Migration

POLM2 - Efficient Application Objects Lifetime Profiler

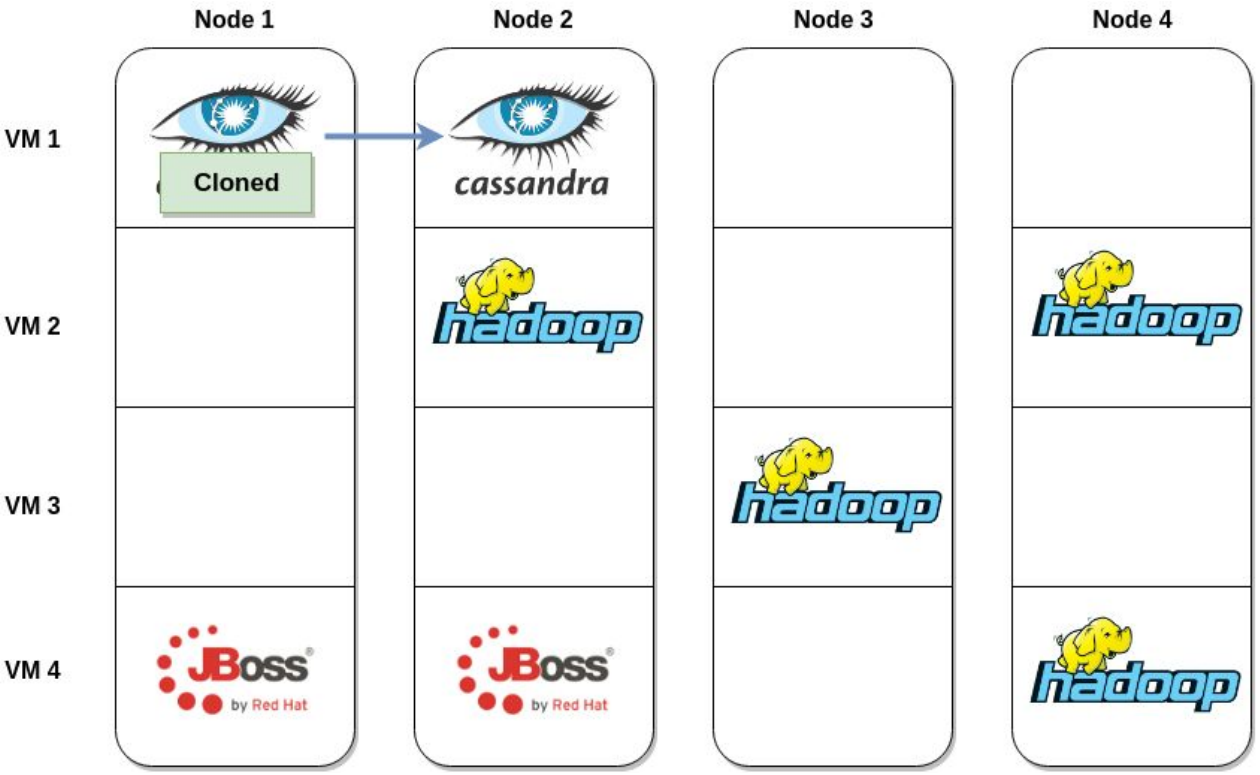
JVM Live Migration (real scenario)



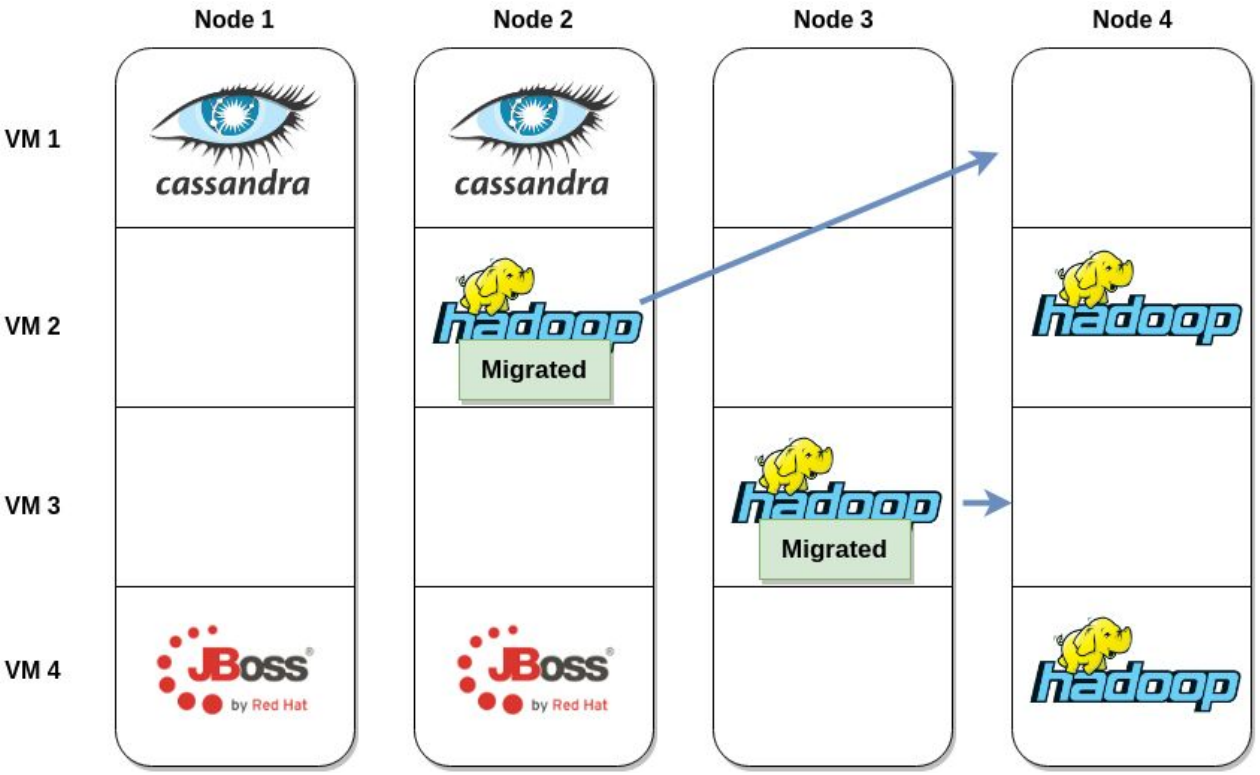
JVM Live Migration (real scenario)



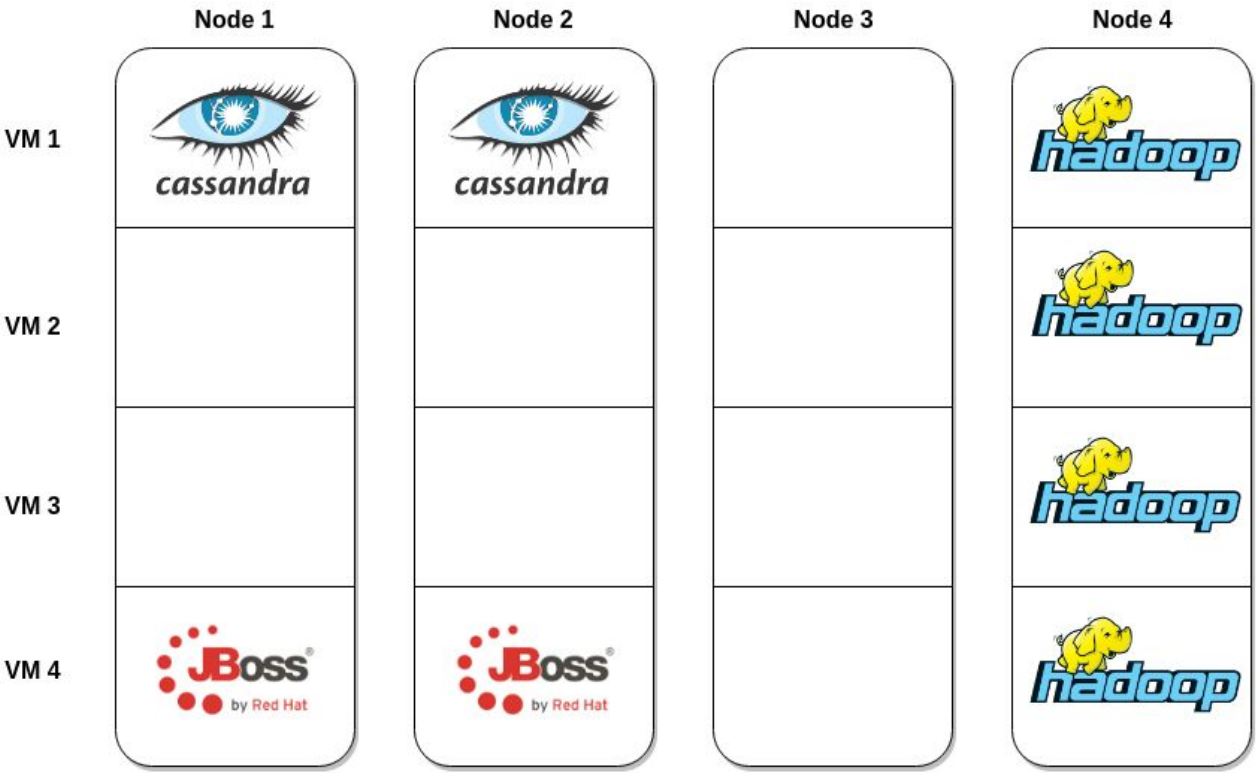
JVM Live Migration (real scenario)



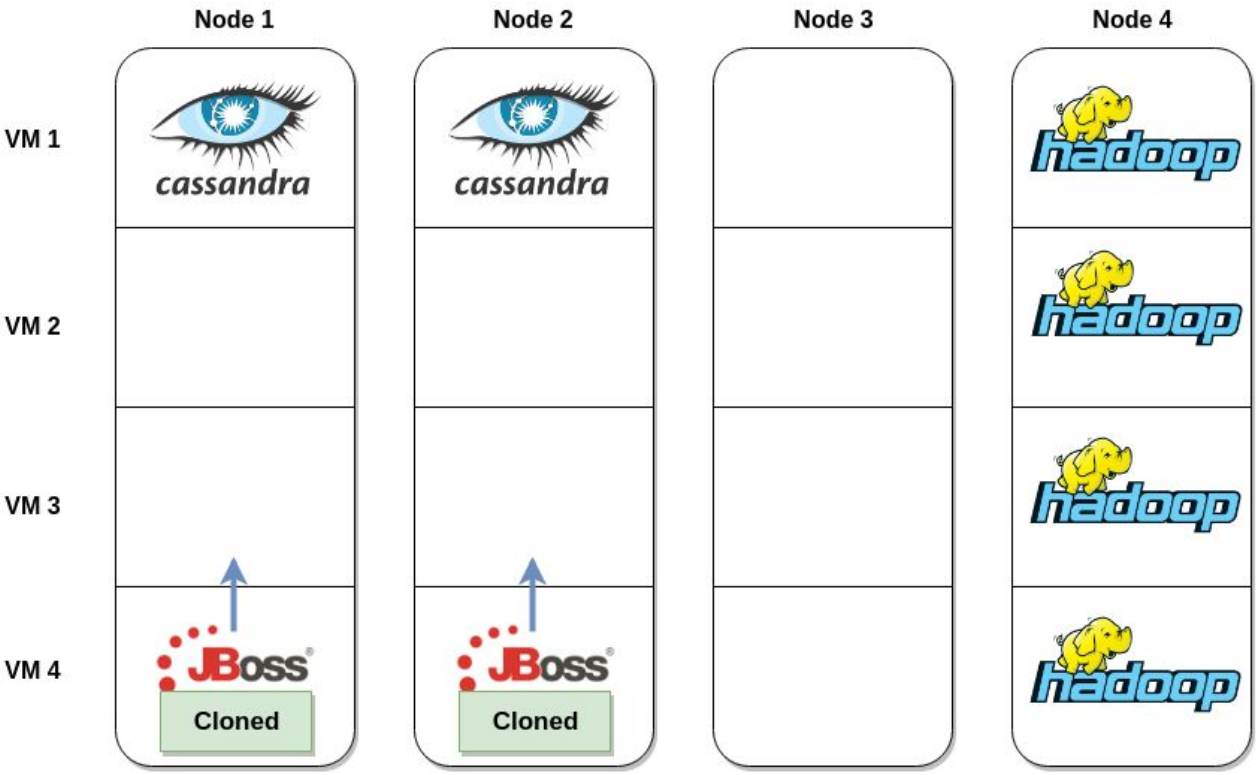
JVM Live Migration (real scenario)



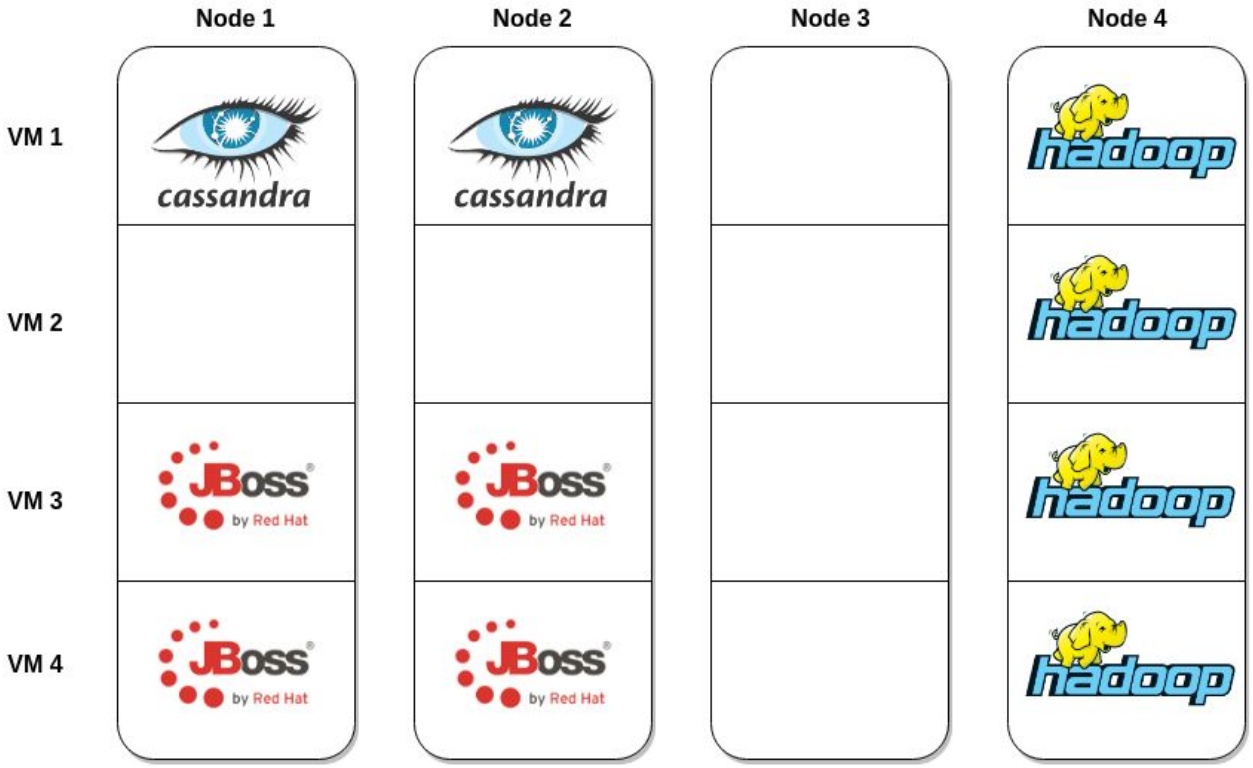
JVM Live Migration (real scenario)



JVM Live Migration (real scenario)



JVM Live Migration (real scenario)



Application Live Migration

- Migration should be as fast as possible
  - Small snapshots

- Many platforms support system-vm live migration
  - many processes plus the kernel are migrated

- Even if only one JVM is migrated parts the memory space are unnecessary
  - Memory locations with no live objects which are kept by the runtime environment.

ALMA - Key Insights

- Migrate only the process (JVM)
  - avoid kernel, other processes, etc;
- Use GC to reduce the snapshot size;
- Dynamically minimize the size of the memory to migrate
  - migrate only live objects
  - only collect regions which can be collected faster than transmitted through the network.

This leads to small (with almost only live data) snapshots.

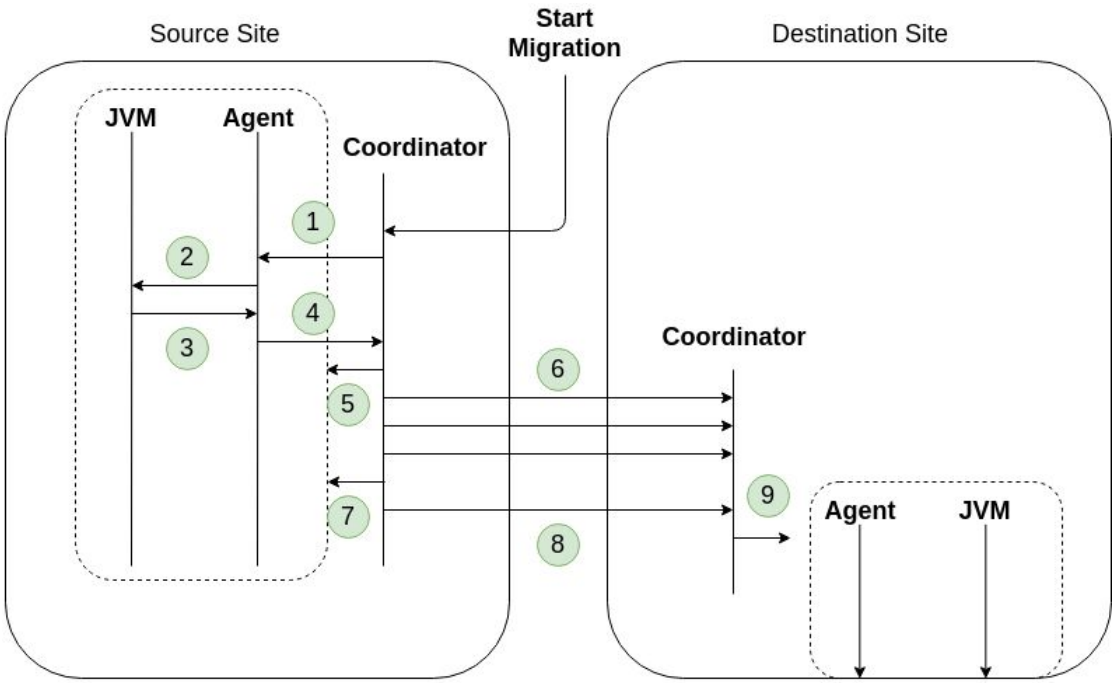
ALMA - Collection Set

Collect regions (memory blocks) that can be collected faster than transmitted through the network:

- Without collection, migration cost is X
- With collection, migration cost is X' + GCCost

X > X' + GCCost

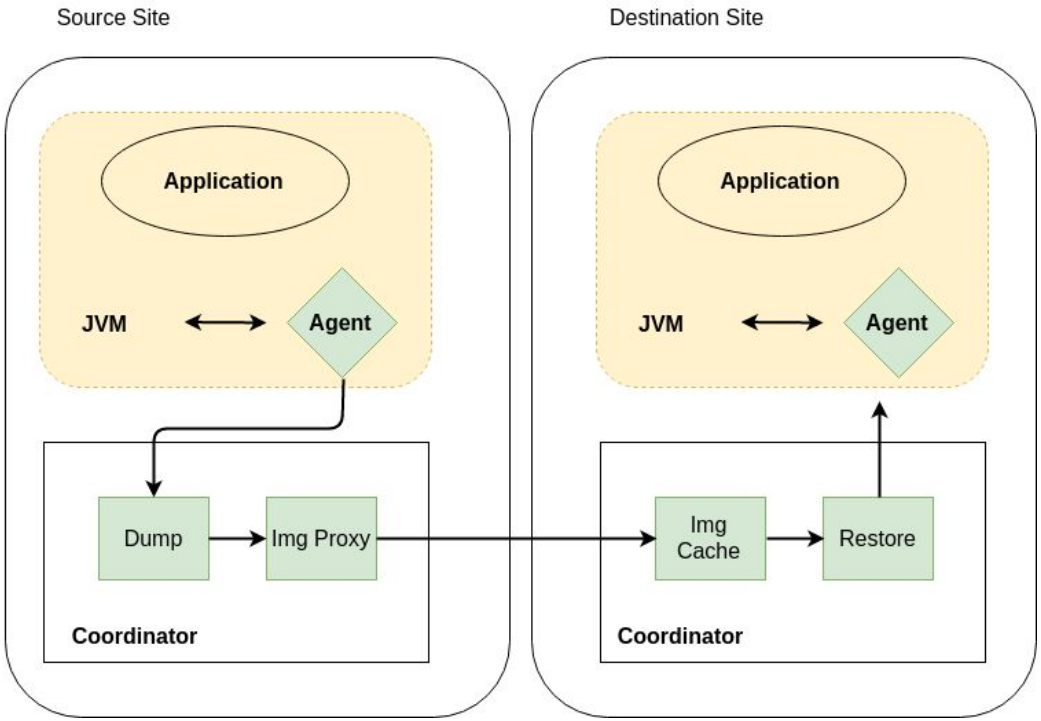
ALMA - Migration Workflow



Steps:

1. Prepare Snapshot
2. Build and Collect CS (Migr. Aware GC)
3. Return Free Mappings
4. Send Free Mappings to Coordinator
5. Checkpoint JVM
6. Send Snapshot
7. Stop JVM, incremental snapshot
8. Send final snapshot
9. Restore JVM from snapshot.

ALMA - Architecture



Components:

- **Application**: target application to migrate;
- **Agent**: analyzes the JVM;
- **Coordinator**: coordinates migration;
- **Dump**: takes JVM snapshots;
- **Img Proxy**: sends snapshot;
- **Img Cache**: caches snapshot;
- **Restore**: restores JVM from snapshots;

ALMA - Implementation

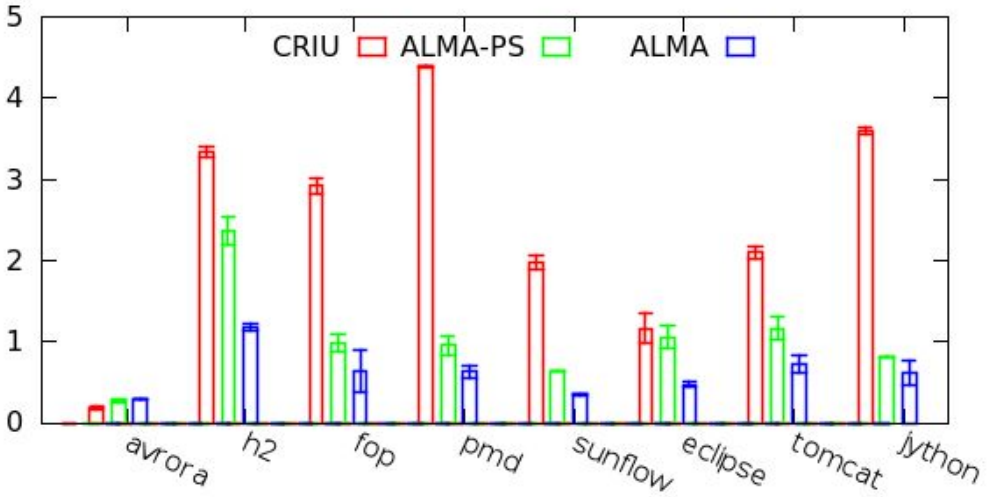
- ALMA augmented HotSpot 8 to support Migration Aware GC;
- Coordinator is implemented by extending CRIU to support remote migration. ALMA added two new components to CRIU:
  - Image Proxy - sends snapshot to the destination site;
  - Image Cache - caches snapshot in the destination site;

ALMA - Evaluation

- Evaluate ALMA's performance compared to:
  - **CRIU** - Checkpoint and Restore for Linux;
  - **JAVMM** (Hou et. al, 2015) - Extends Xen to migrate Java applications. It simply collects the young generation before migration;
  - **ALMA-PS** - Similar to JAVMM but based on CRIU.
- Environment:
  - OpenStack VMs with 4vCPUs and 4GB RAM
  - DaCapo and SpecJVM2008 benchmark suites

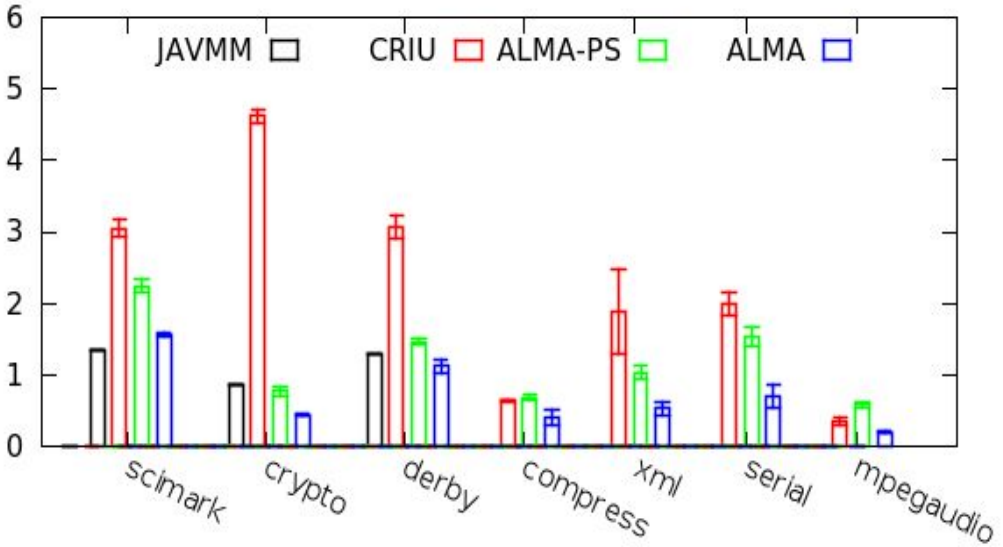
Evaluation - Application Downtime (seconds)

DaCapo



The Smaller the Better!

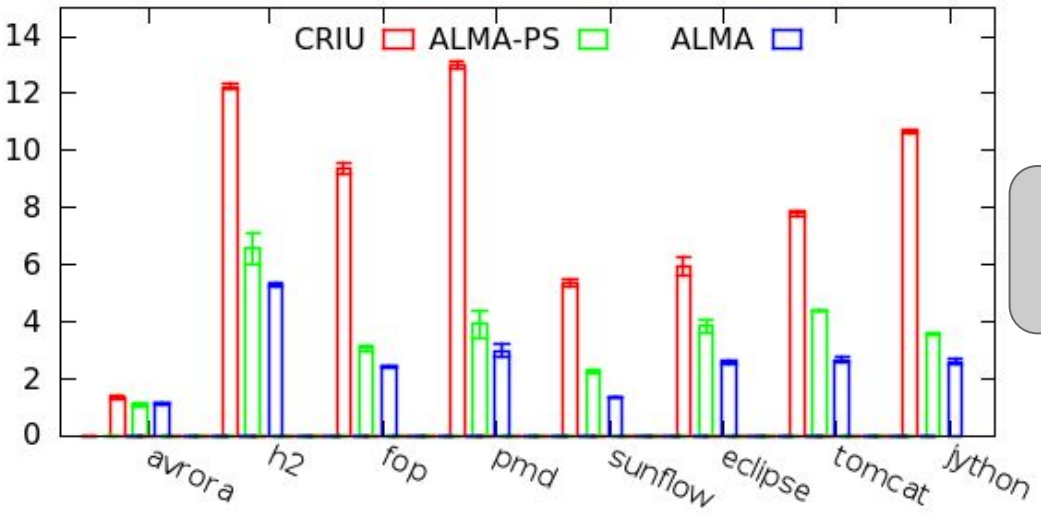
SPECjvm2008



The Smaller the Better!

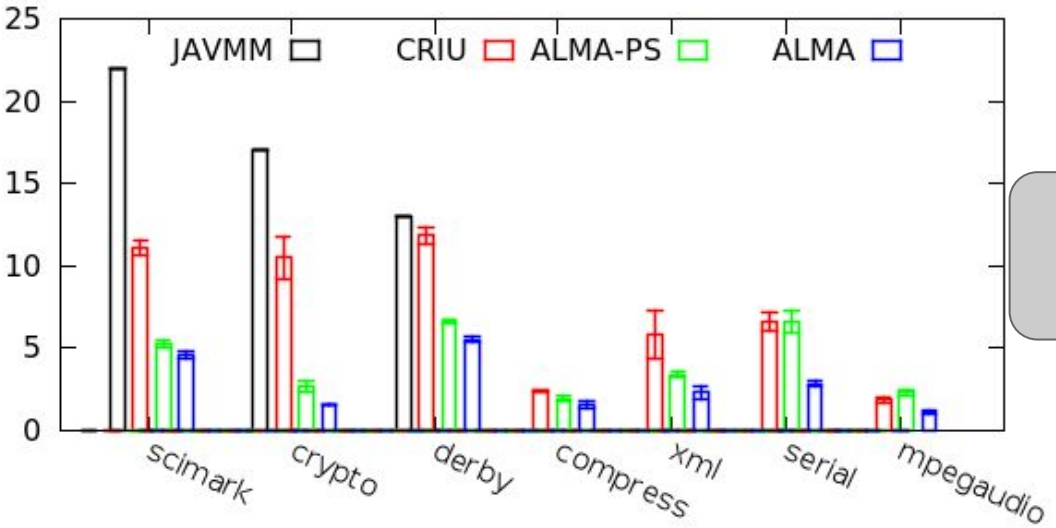
Evaluation - Total Migration Time (seconds)

DaCapo



The Smaller the Better!

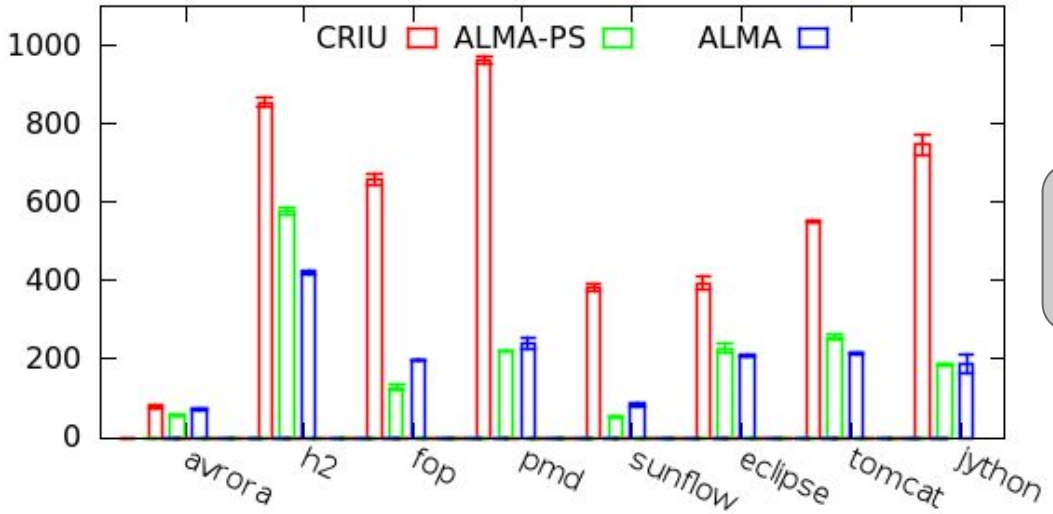
SPECjvm2008



The Smaller the Better!

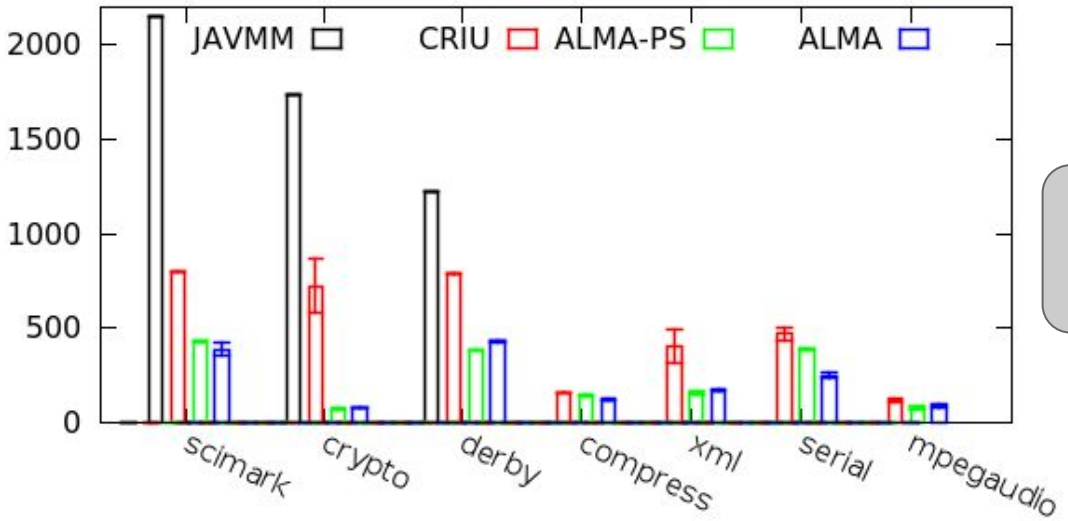
Evaluation - Network Bandwidth Usage (MBs)

DaCapo



The Smaller the Better!

SPECjvm2008



The Smaller the Better!

ALMA - Summary

- ALMA offers efficient migration of Java server applications
  - by selectively avoiding garbage when it pays off
  - by ignoring unmodified memory pages
  - by migrating only the target process
  - without requiring changes to applications
    - Only the agent and runtime are changed
- ALMA's implementation is based on OpenJDK and CRIU;
  - Code is available at: github.com/rodrigo-bruno/alma
- ALMA outperforms current solutions in all evaluated metrics

TÉCNICO LISBOA

gsd inesc id lisboa

ALMA - ALMA - JVM Live Migration

POLM2 - Efficient Application Object Lifetime Profiler

JVM Memory Profiling

- Why?
  - Analyzing memory leaks in GC managed languages
  - Analyze application allocation profiles
    - Object lifetimes, size, …
    - Etc…
- Most profilers either sample memory or do full heap dumps (using *jmap)* resulting in
  - Uncomplete profiling data (only consider a sample)
  - Massive performance overhead due to full heap dumps

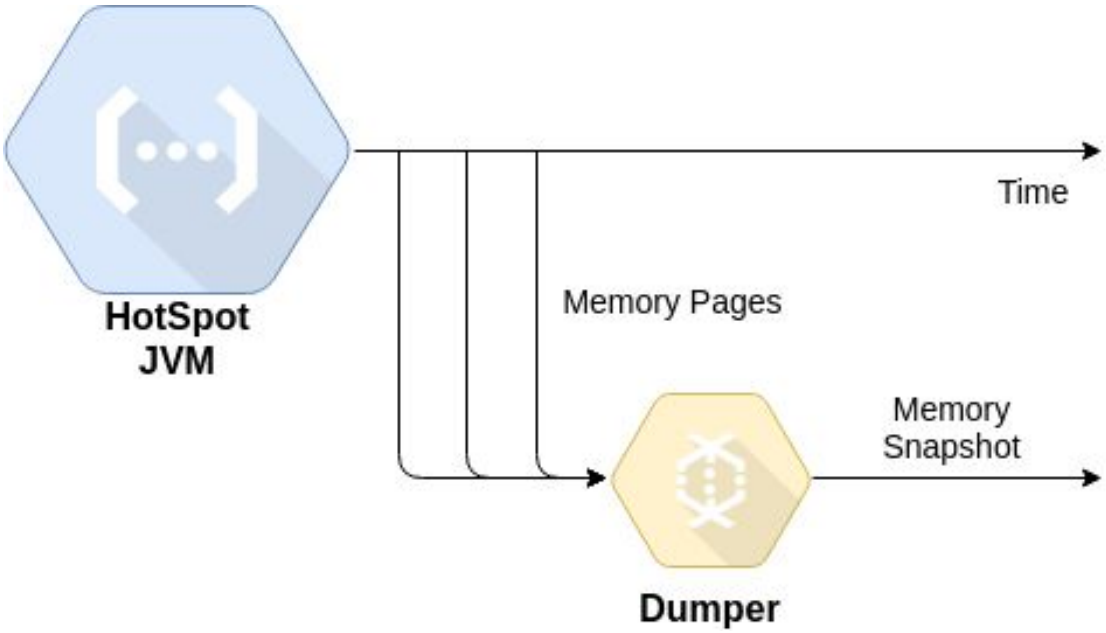
POLM2 - Key Insights

- Create full heap dumps
  - That can be analyzed offline

- Heap dumps are incremental
  - Do not include unmodified memory pages

- Heap dumps avoid garbage
  - Pages that contain no live objects

Efficient Application Object Lifetime Profiler



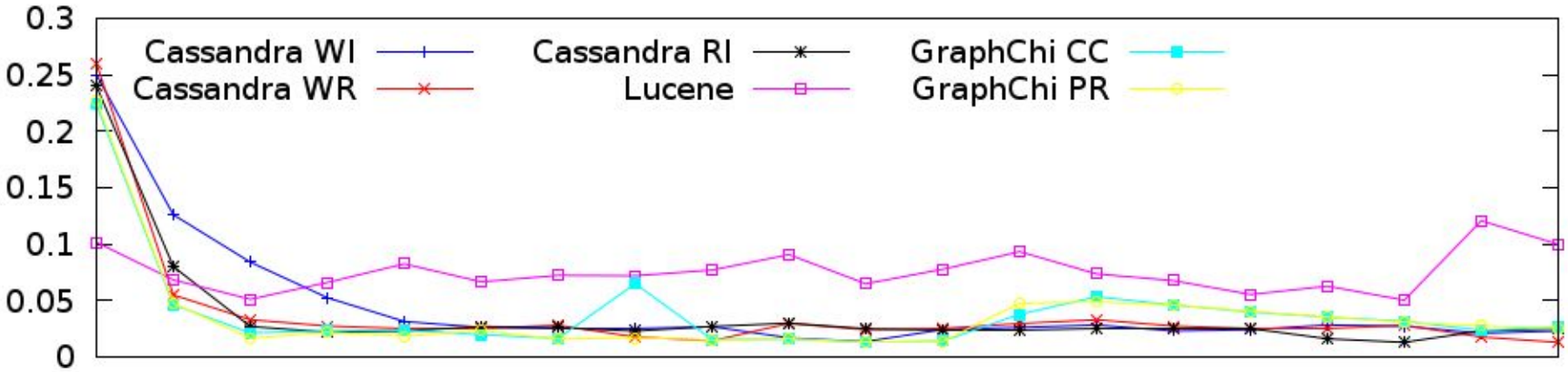
POLM2 - Implementation

- CRIU is used to implement the Dumper component
- CRIU is able to iteratively create snapshots
  - Which result in very fast incremental snapshots
- OpenJDK patched to use MADVISE syscall to mark as "NO_NEED" pages that contain no live objects
  - Pages are marked after each GC cycle
  - CRIU ignores this pages
    - Resulting in a shapshot that contains only live objects

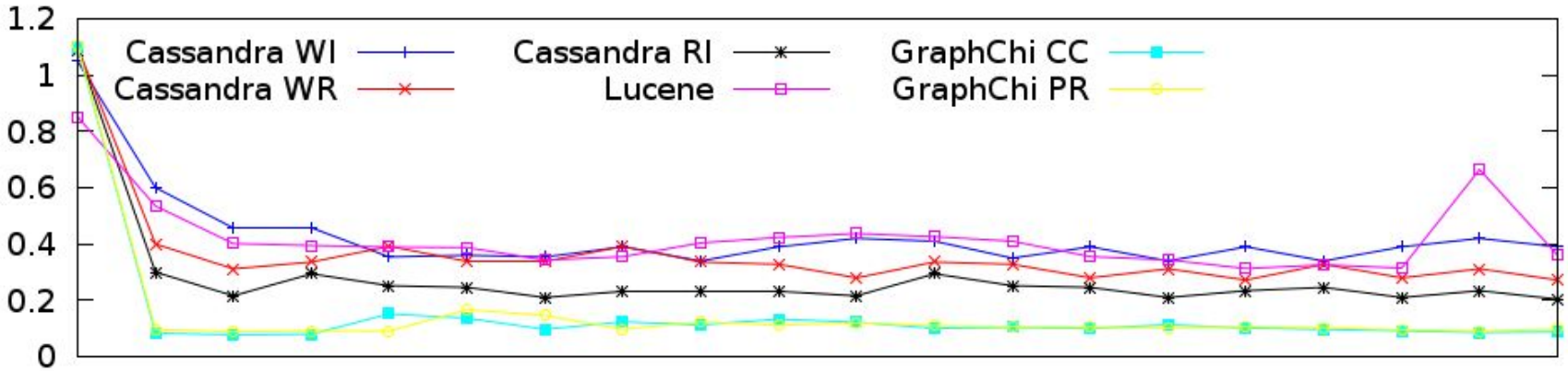
POLM2 - Evaluation

- Compare POLM2 snapshots (using CRIU) with *jmap*
- Big Data Platforms & Workloads:
  - Cassandra (**Key-Value Store**)
    - YCSB workloads
  - Lucene (**In-Memory Indexing Tool**)
    - Read/Write transactions on Wikipedia dump
  - GraphChi (**Graph Processing Engine**)
    - Twitter graph dump (42M vertexes, 1.5B edges)
      - PageRank
      - Connected Components

Evaluation - Snapshot Time (norm to *jmap*)



Evaluation - Snapshot Size (norm to *jmap*)



Summary

- C/R is a powerful tool that has many interesting use-cases
  - Efficient runtime-aware migration
  - Efficient runtime-aware application profiling
- C/R at process level allows more fine grained control over what resources are being migrated
- Open problems:
  - Security - efficiently move processes/containers with sensitive data
  - Local Data - efficiently move local resources such as files
  - ...

Thank you for your time.
Questions?

Rodrigo Bruno
email: rodrigo.bruno@tecnico.ulisboa.pt
webpage: www.gsd.inesc-id.pt/~rbruno
github: github.com/rodrigo-bruno