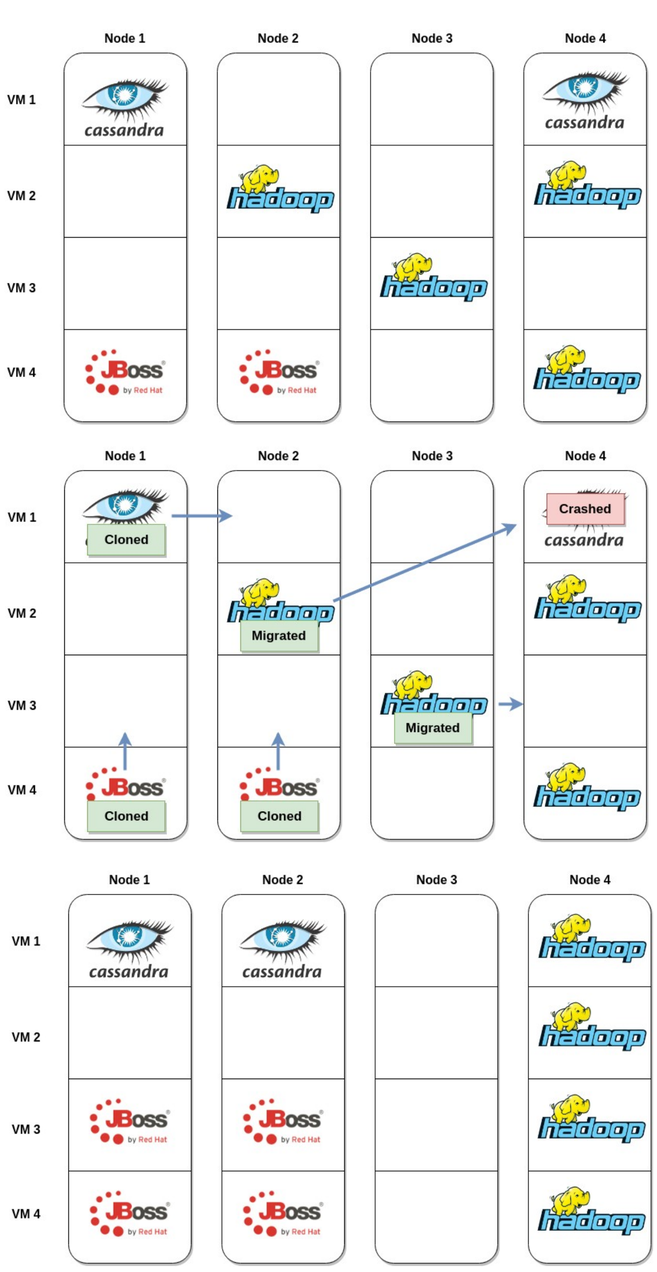
ALMA – GC-assisted JVM Live Migration for Java Server Applications

Rodrigo Bruno, Paulo Ferreira – rodrigo.bruno@tecnico.ulisboa.pt, paulo.ferreira@inesc-id.pt

1. Problem



2. Solution

- Migrate **only the JVM** (avoid kernel and other processes);
- **Avoid unreachable data** (garbage) in the snapshot

  by selectively collecting memory regions that would take longer to transmit than to collect.

3. Other Approaches

- Force application throttling;
- Rely on high-speed networks;
- Force full GC before migration;
- Fail to determine the live working set;
- Migrate a system-VM when only a process is needed;
- Do not avoid unreachable data.

4. ALMA – Migration-Aware GC

$$Data = \sum_{Heap} used(r) - \sum_{CS} dead(r)$$

Size of Snapshot

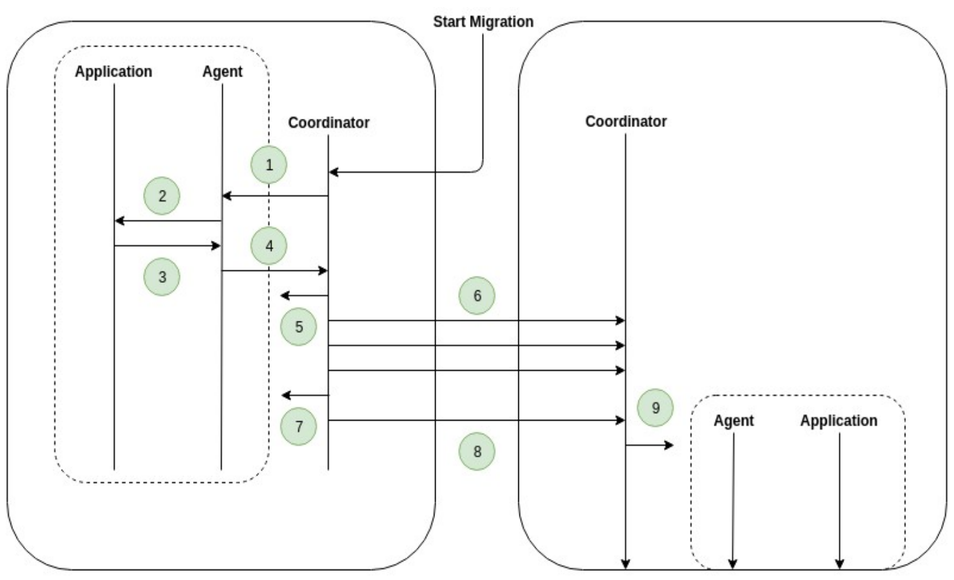
$$GCRate(r) = \frac{dead(r)}{cost(r)}$$

Collector Rate (space per amount of time)

$$CS = \{\forall r : GCRate(r) > NetBandwidth\}$$

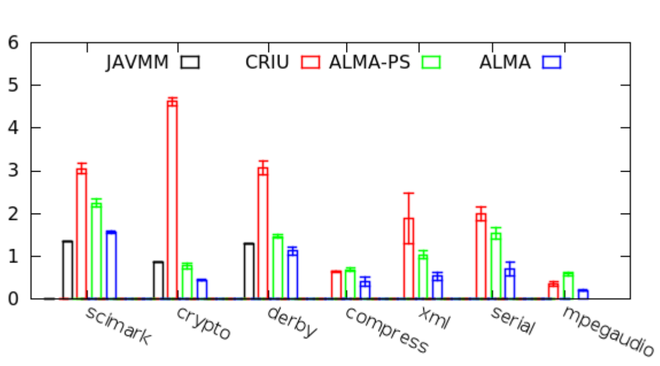
All regions which can be collected faster than transmitter, are selected for collection!

5. ALMA – Migration Workflow

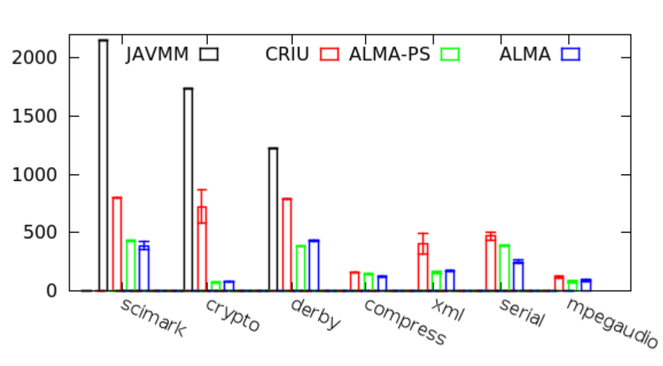


1. Prepare Snapshot
2. Mirg. Aware GC
3. Return Free Maps.
4. Send Free Maps.
5. Checkpoint JVM
6. Send Snapshot
7. Incr. Snapshot
8. Send Final Snapshot
9. Restore JVM

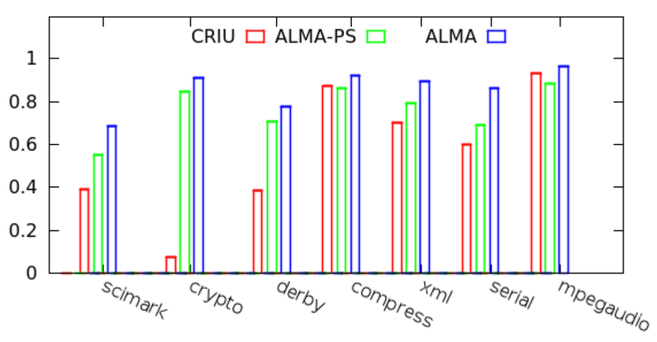
6. Results



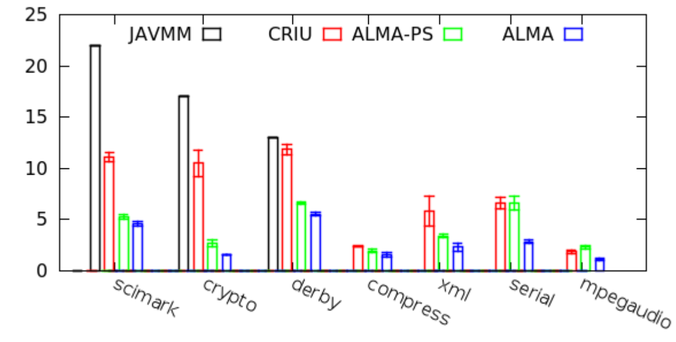
Application Downtime (seconds)



Network Bandwidth Usage (MBs)



Application Throughput (normalized)



Total Migration Time (seconds)

For more information, please contact us or checkout ALMA at github.com/rodrigo-bruno/alma