Imperial College London

LSDS
Large-Scale Data & Systems Group

ORC: Increasing Cloud Memory Density via Object Reuse with Capabilities

Vasily A. Sartakov
Imperial College London
http://lsds.doc.ic.ac.uk
<v.sartakov@imperial.ac.uk>

Joint work with **Lluís Vilanova**[1], **Munir Geden**[1], **David Eyers**[2], **Takahiro Shinagawa**[3], **Peter Pietzuch**[1]

[1]**Imperial College London**, [2]**University of Otago**, [3]**The University of Tokyo**

USENIX OSDI – July 2023

Duplicate Cloud Software Components

☛ VMs/containers have **similar memory content**

Virtual machines have their own **guest OS kernel**

- Usually the same across VMs

Virtual machines and containers run **similar applications**

- E.g. many users deploy the same NGINX web server

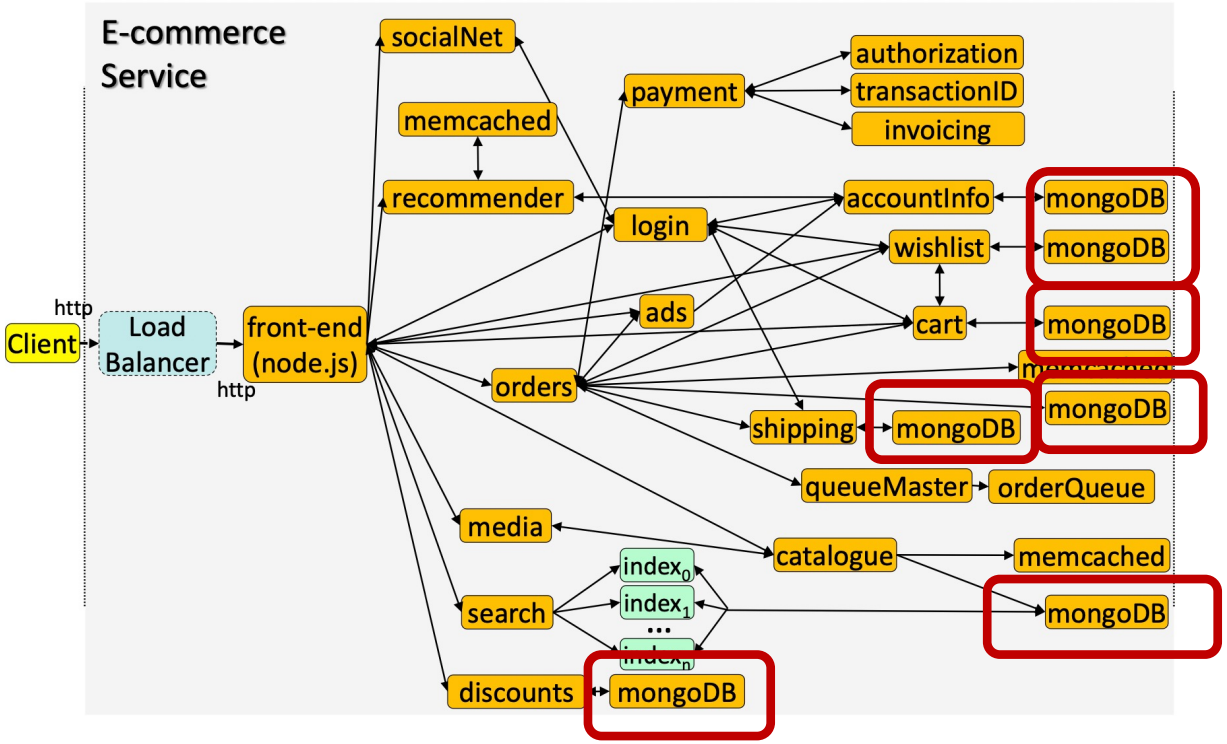
Different applications are often built on top of the **common frameworks**

- E.g. use the same Python runtime
- E.g. many dynamic libraries will be the same (*lib*.so*)

Duplicate Components in Microservices

DeathStarBench microservice benchmark:

- Many duplicate mongoDB instances
- Each mongoDB instance uses 640 MB



source: Y. Gan et al., ASPLOS 2019

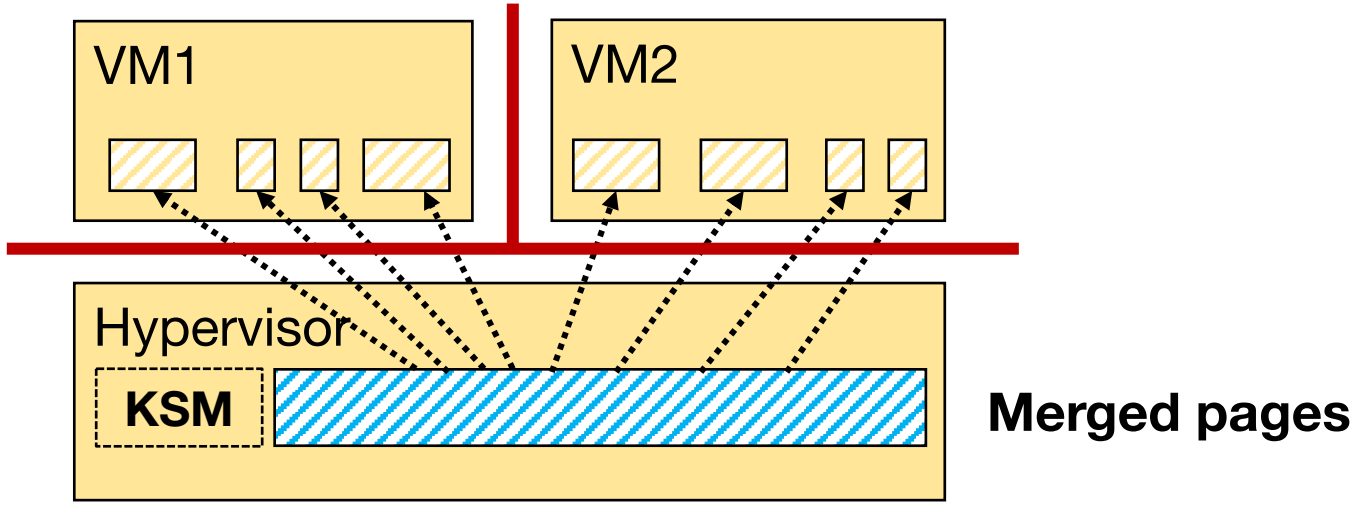
Large binaries:
518 MB total

```
148M Nov  5  2021  mongosh
104M Dec 19  2013  mongod
72M Dec 19  2013  mongos
56M Dec 19  2013  mongo
17M Oct 12  2021  mongofiles
17M Oct 12  2021  mongorestore
16M Oct 12  2021  mongodump
16M Oct 12  2021  mongoimport
16M Oct 12  2021  mongoexport
16M Oct 12  2021  mongostat
16M Oct 12  2021  mongotop
14M Oct 12  2021  bsondump
3.4M Oct 19  2020  perl
3.4M Oct 19  2020  perl5.30.0
1.2M Jun 18  2020  bash
1.1M Jan  6  2021  gpg
875K Jan  6  2021  gpgcompose
736K Aug 23  2021  openssl
```

Duplicate memory:
x7 of each execution → 85%

SOTA: Kernel Same Page Merging (KSM)

Hypervisor-led memory deduplication done by Linux kernel



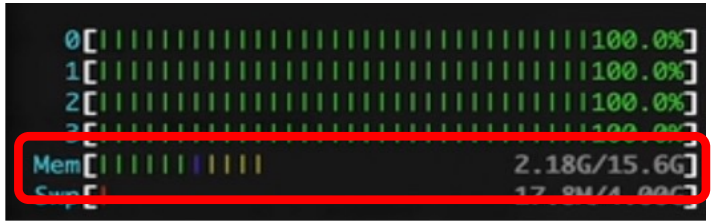
1. KSM periodically scans memory pages and calculates their hashes
2. KSM removes duplicate pages by remapping pages for sharing

☛ Many tuning parameters that govern KSM operation

Issues with KSM

KSM is **probabilistic in nature**
- Deduplication benefit unpredictable

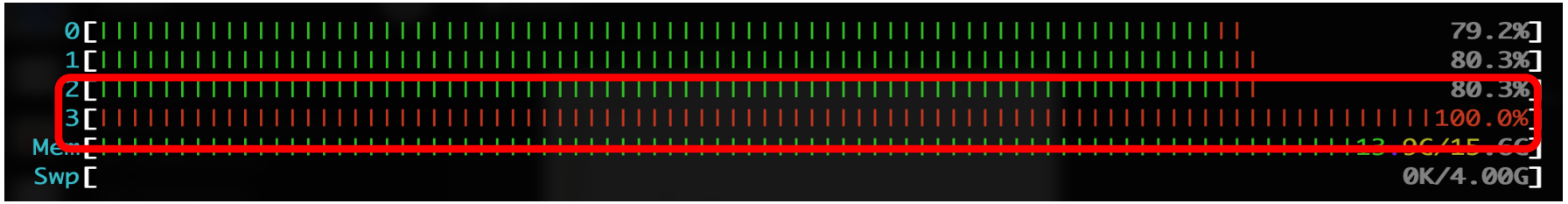
✓ **First KSM run:** memory reduces 13 GB ➔ **2.2 GB**



KSM **consumes CPU cycles**
- Deduplication adds substantial overhead

✗ **Next KSM run:** memory reduces 13 GB ➔ **11 GB**

Active KSM: 100% CPU



Only cloud provider benefits from KSM
- Tenants unaware what is de-duplicated and when

Hypervisor and MMU Tax

Hypervisor uses MMU to isolate VMs

- Page manipulation → performance and tail-latency overhead

Hypervisor emulates HW platform

- Lacks the visibility that an OS has in terms of application-level load-time semantics → page scanning

MMU operates at page granularity

- COW mechanisms vulnerable to side-channel attacks

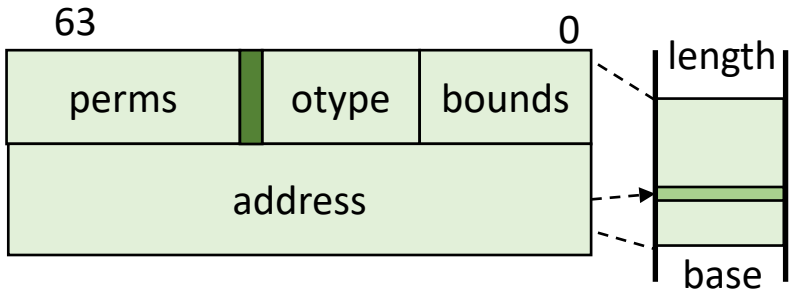
Can we use another technology for isolation and deduplication?

👉 **CHERI: low-overhead capability-based compartments**

Background: CHERI Capabilities

Fat pointers protected by hardware:

- base + length, cursor
- permission, tag
- byte-granularity*



Fine-grained isolation

Limited dependency on OS kernel

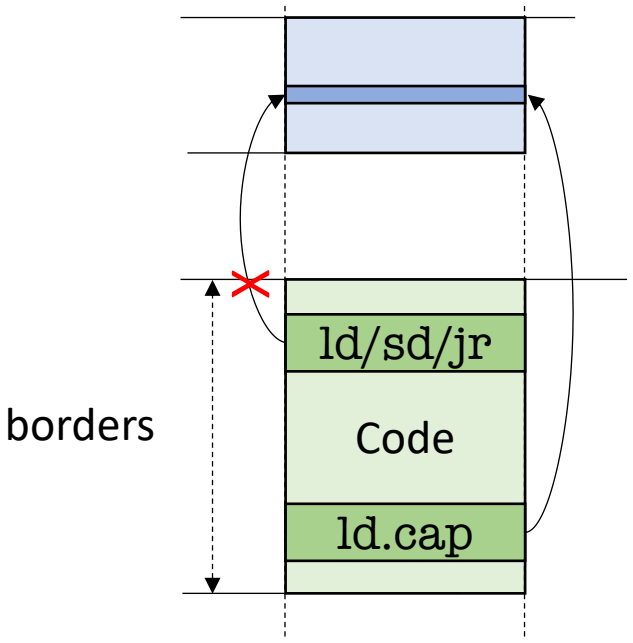
Available: Arm CHERI Morello Boards (Armv8)

Capabilities can be created only from capabilities

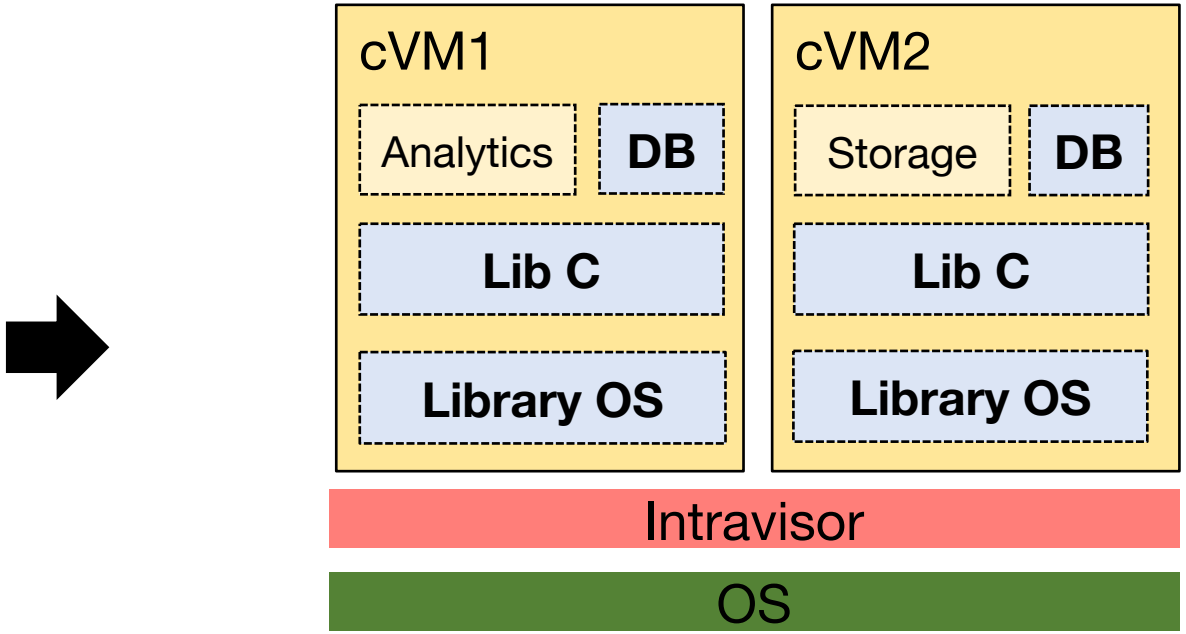
- Using cap-aware instructions, but not a privileged intermediary

Background: Cap-based Compartments

Capabilities with the same bounds create an isolated compartment



Cap-based compartment

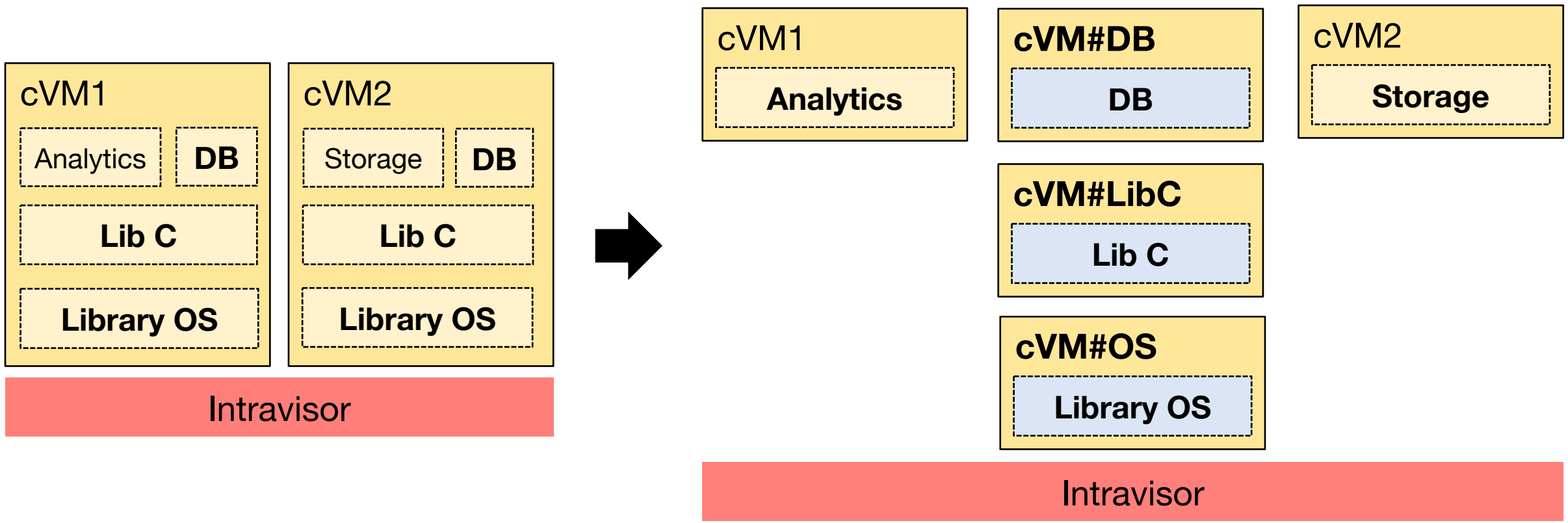


Cloud stack built on top of cap-based compartments

- Lightweight isolation and sharing
- Shared single address space → Control over components

ORC: Object Reuse with Capabilities

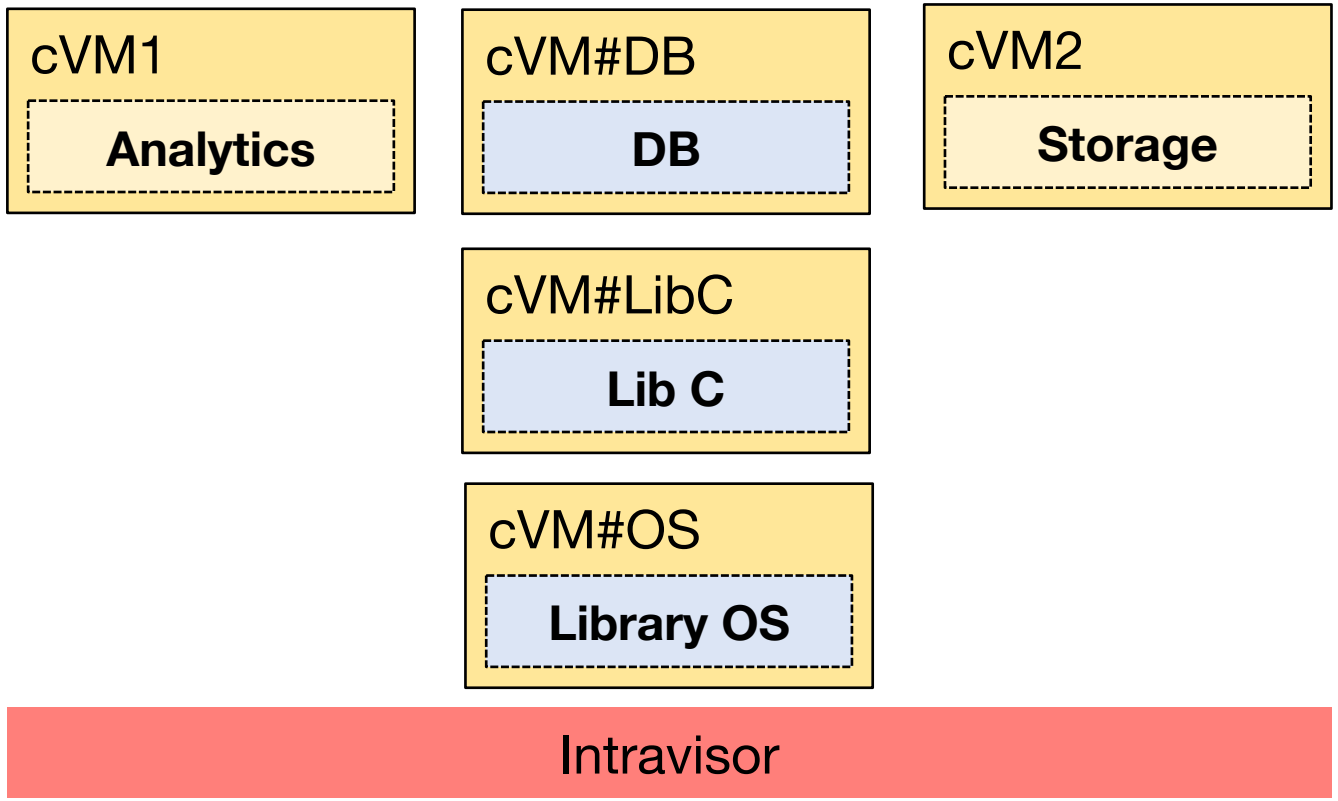
Key idea: Decompose into components with **binary sharing**



Challenges: ORC

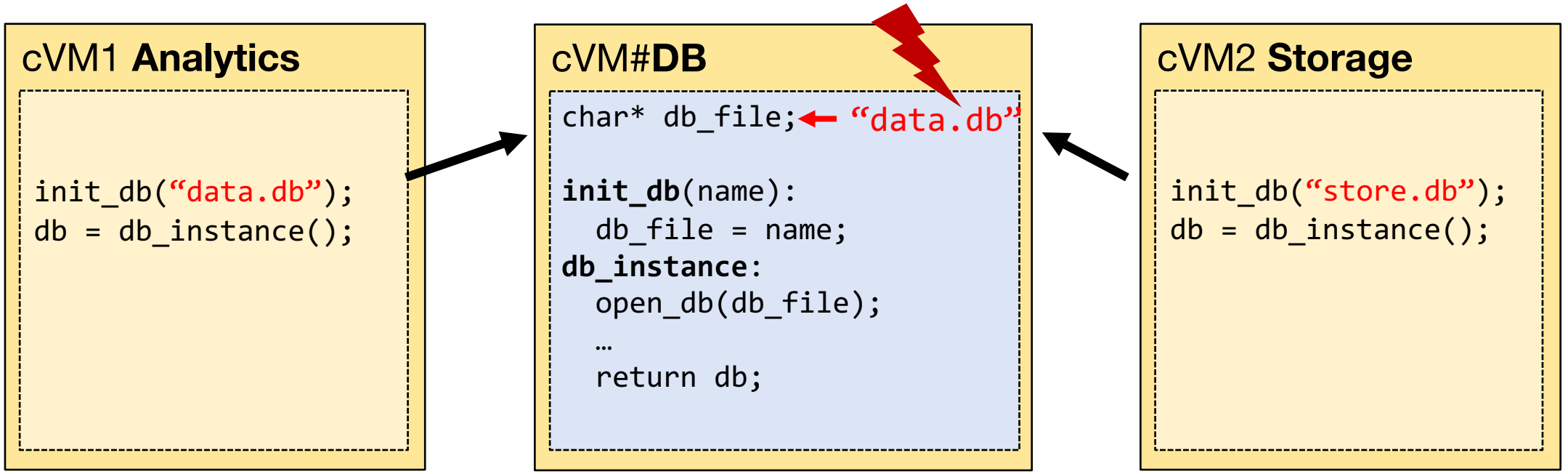
1. Safe sharing of **immutable & integrity-protected objects**

2. Keep **Intravisor TCB small** when loading shared objects



1. Safe Sharing of Immutable Objects

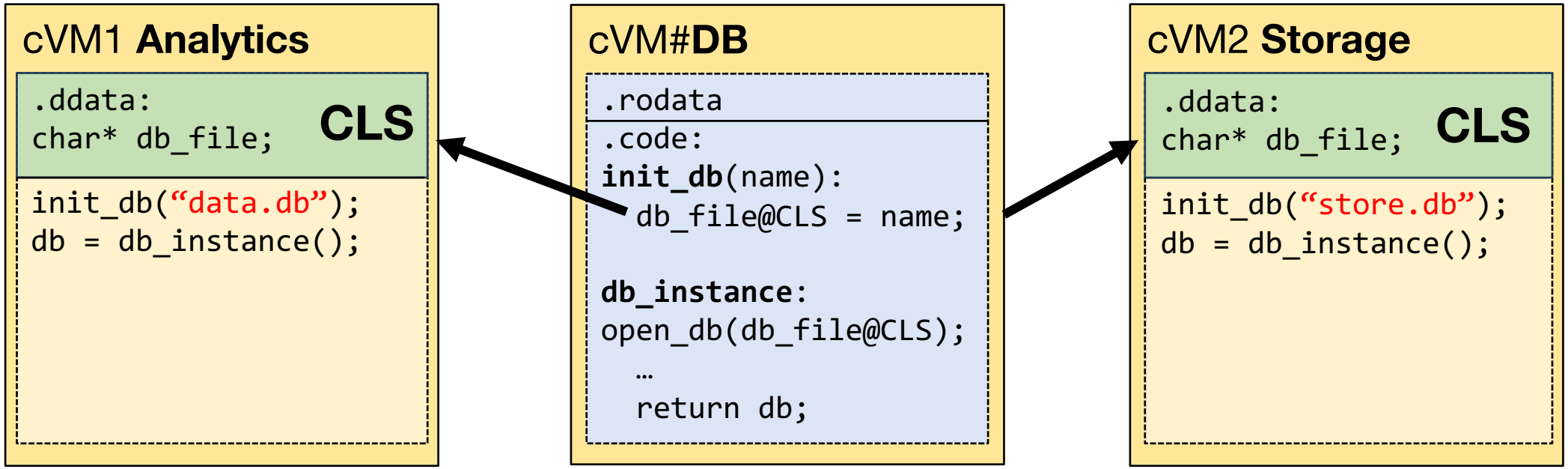
Shared components have **state**, e.g. global variables



☛ Need to duplicate global variables to make shared objects immutable

Compartment-Local Storage (CLS)

CAP-VMs have private copy of globals of shared objects

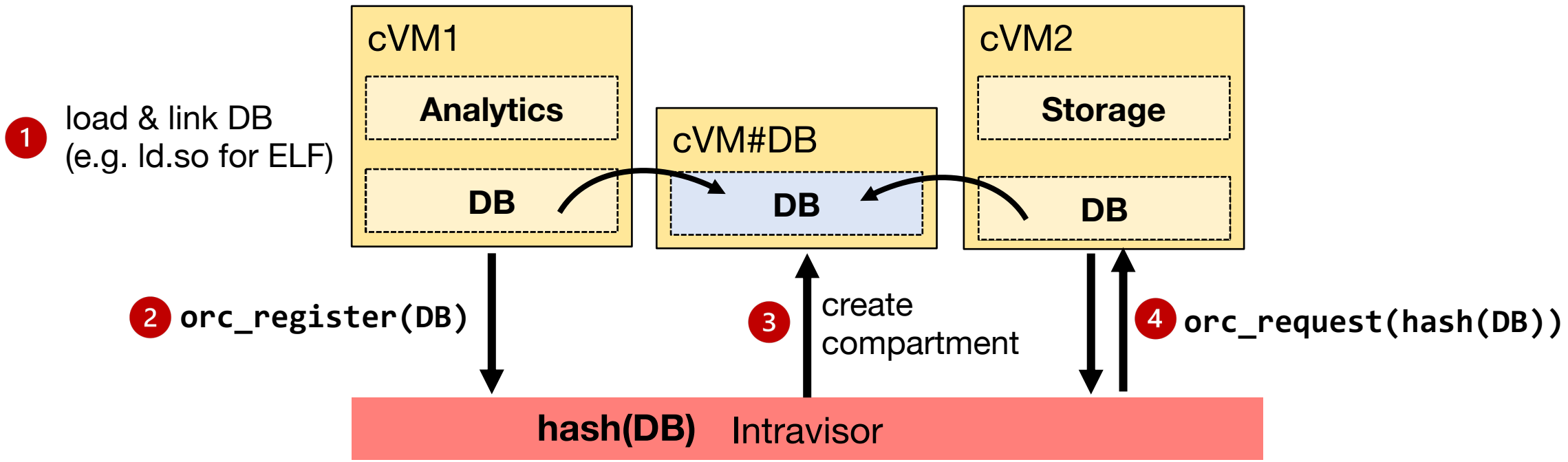


Implemented as **LLVM compiler pass** using memory capabilities

- Similar to thread-local storage (TLS)
- Compiler adds calls to `__cls_get_addr()` function

2. Small TCB with Shared Objects

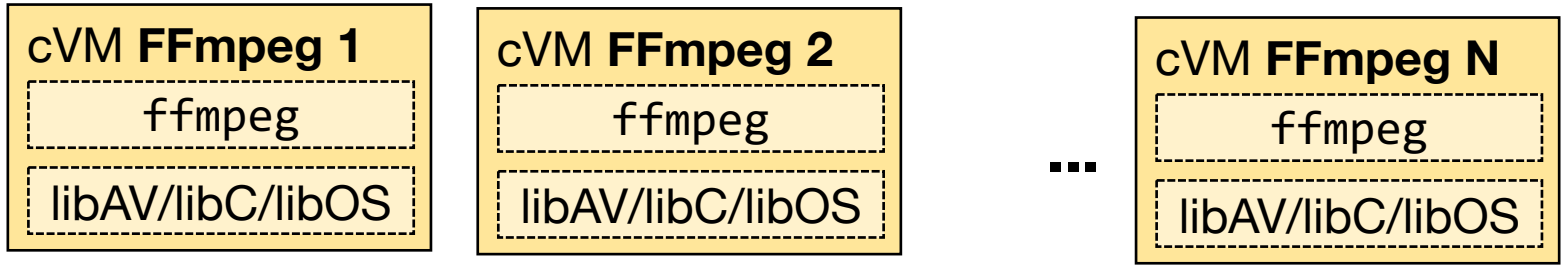
Idea: Shared object loading is done by an **untrusted loader**



Evaluation: Video Transcoding Service

Cloud-based **video transcoding service:**
`ffmpeg` video transcoder + libraries + library OS

With **KSM**

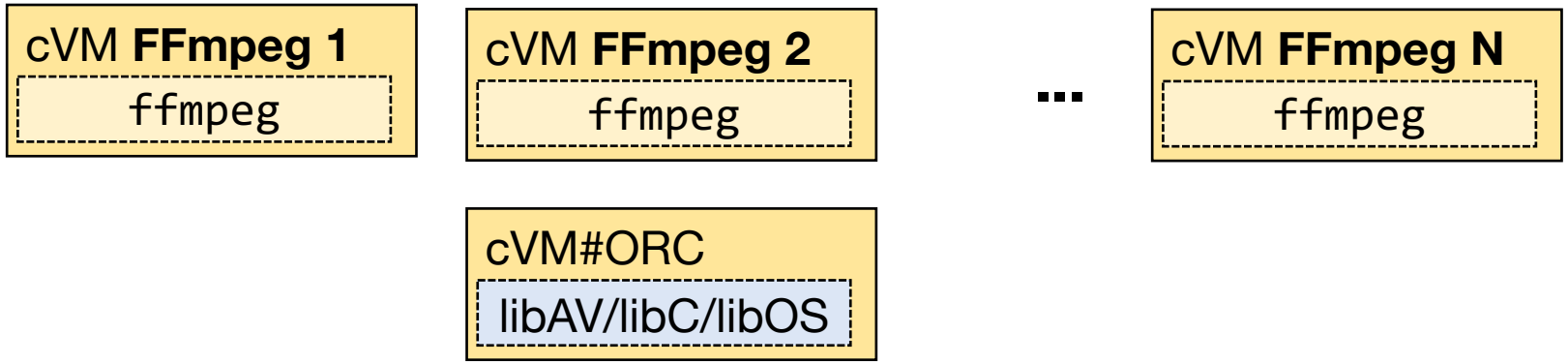


Memory per `ffmpeg` worker: **110 MB**

Launches new `ffmpeg` instance per request

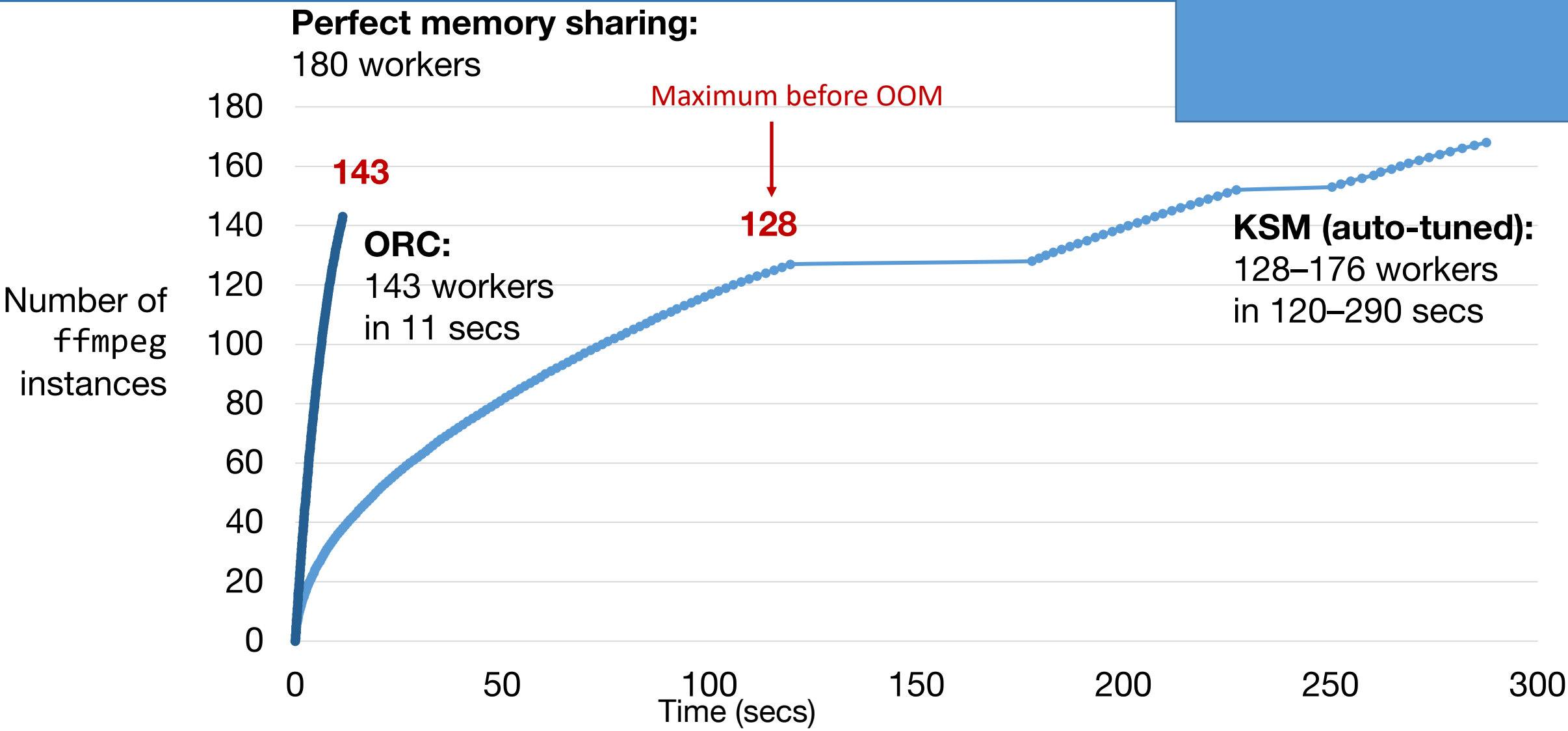
– Limited by memory or CPU time

With **ORCs**



ORC-shareable memory: **11 MB**

ORC vs. KSM on Arm Morello



☛ ORC deduplicates immediately while finding less duplicate memory

Conclusions: Object Reuse With Capabilities

Many VMs/containers have similar memory content

MMU and hypervisor-based solution is not efficient

- Scanning/deduplication overhead
- Not controlled by a tenant
- Probabilistic in nature

ORC: **Memory de-duplication** with semantic **object reuse and capabilities**

- Sharing by design
- Object reuse with preserved state
- Deprivileged policy/mechanisms fully controlled by a tenant

Thank you! – Any Questions?

Vasily A. Sartakov v.sartakov@imperial.ac.uk

Source code: http://github.com/lsds/intravisor

UK Research and Innovation

Technology Innovation Institute

This work was funded by the Technology Innovation Institute through its Secure Systems Research Center, and the UK Government's Industrial Strategy Challenge Fund under the Digital Security by Design Programme (UKRI grant EP/V000365 "CloudCAP"). This work was also supported by JSPS KAK- ENHI grant number 18KK0310 and JST CREST grant number JPMJCR22M3.