CMSBench: Open, Repeatable Benchmarking of Far Memory

Grant Mackey
CTO Jackrabbit Labs

- There isn’t an industry consensus on the utility of far memory today
- “Show, don’t tell” benchmarks are useful
- Make it easy, make it digestible
- There isn’t a de-facto effort in the wild yet to tackle this challenge
- The OCP Composable Memory Systems group is taking a crack it it
  - https://github.com/opencomputeproject/OCP-SVR-CMS-Benchmarks

- There isn’t an industry consensus on the utility of far memory today
  - Not a lot of ecosystems work, still seen as fledgling or niche
- “Show, don’t tell” benchmarks are useful
  - Demonstrate utility with a suite of known end-user applications is high value
- Make it easy, make it digestible
  - Benchmarking is hard, hard means people don’t do it or don’t do it well
- There isn’t a de-facto effort in the wild yet to tackle this challenge
  - What’s important, what’s reasonable, etc. are unsettled questions
- The OCP Composable Memory Systems group is taking a crack it it
  - https://github.com/opencomputeproject/OCP-SVR-CMS-Benchmarks

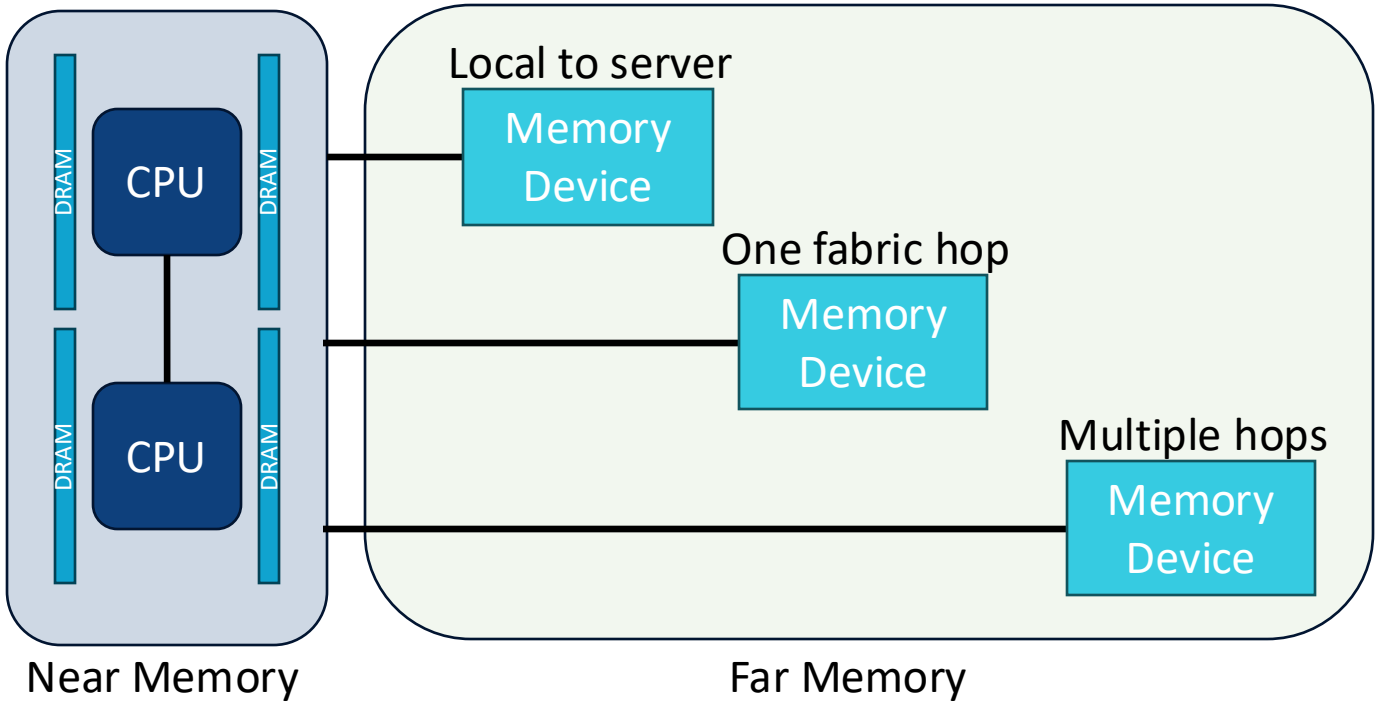
Introduction

What is the OCP[1] Composable Memory System group?

- Kicked off in 2022, one of the few software-first subgroups in OCP
  - Focused on architecture and systems, rather than hardware and devices
- Workstreams
  - *Composable Workloads*
  - Memory Fabric Orchestration
  - AI and Composable Memory
  - Computational Programming
  - Academic Research
- Focus is on far memory technologies and their fabrics in general
  - CXL, UALink, etc. etc.

1. ”wait, what’s OCP?” → click me

- What's your definition of far memory?
  - Let's call it as "not a local DRAM NUMA domain, but still a NUMA node"



Introduction

Why does CMS care about this, what's the utility?

- CMS is composed of groups that either
  - Make far memory devices and fabrics
  - Work on the connectivity of these technologies
  - Want to consume far memory technologies
- There is a gap in getting from hardware to useful/valuable architectures
- The multi-industry group has a large network to leverage to get from standards to practice

The Problem

Benchmarking is hard

- Many benchmarks do not have a good out of box experience.
  - Installation
  - Configuration
  - Comparing A and B
- Value is subjective, what is ‘performance’
  - Bandwidth/Latency?
  - Power/Efficiency?
  - TCO?
- There are a ton of benchmarks, but are they doing the thing you care about?
  - Are you exercising the right thing?
  - Is the workload right?
  - Is the scale right?

The Problem

Independent Replication of Results is Hard

- “I did a thing, and it is better than all things before!” – pick 1 to 2
  - There is a description of the software stack
  - There is a description of the hardware used
  - The code is open source
  - There is a how-to guide on replication
- Even good intentions are hard to replicate
  - Library versions
  - Kernel versions
  - Compiler
  - State of the system during testing, etc. etc.
- Output is non-standard
  - Why are you plain-text, you CSV, and you json-ish?

The Problem

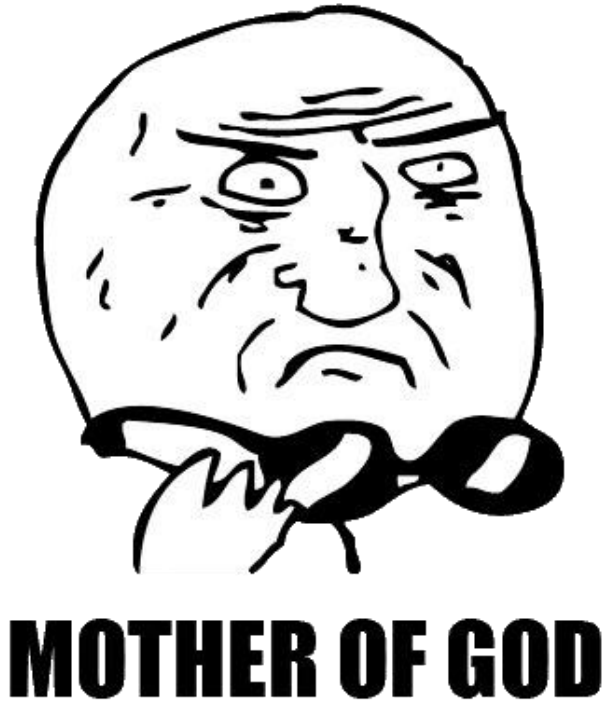
Independent Replication of Results is Hard

- “I did a thing, and it is better than all things before!” – pick 1 to 2
  - There is a description of the software stack
  - There is a description of the hardware used
  - The code is open source
  - ~~There is a how-to guide on replication~~ No there isn’t
- Even good intentions are hard to replicate
  - Library versions
  - Kernel versions
  - Compiler
  - State of the system during testing, etc. etc.
- Output is non-standard
  - Why are you plain-text, you CSV, and you json-ish?

The Problem

Far memory makes this even more convoluted

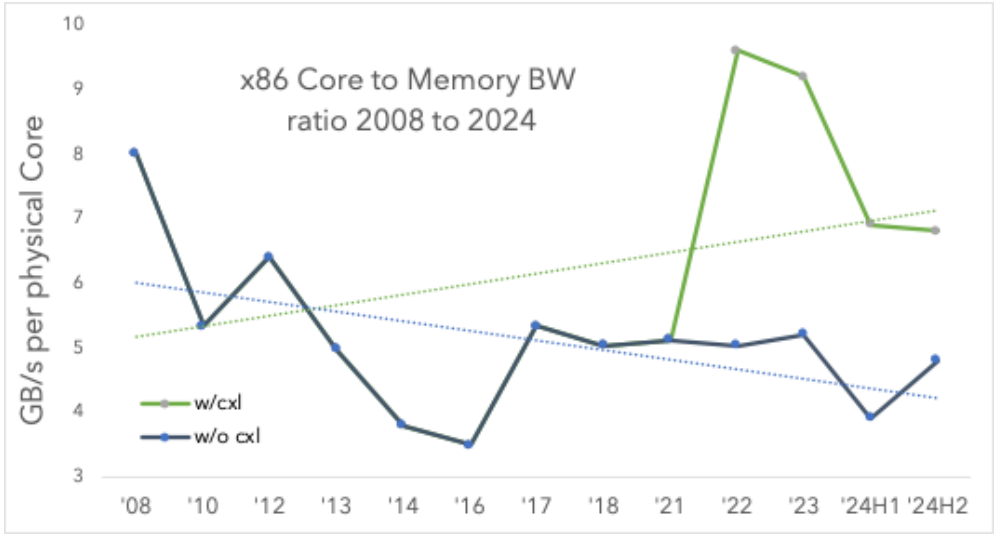
- Standardization means that you can easily mix and match ISAs in a workload
  - Arm/x86, GPUs, RISC-V, xaccelerator
- Fabric topologies don't tell you what they are, just how 'far' something is
- You can tune far memory QoS as a workload is running
- You can dynamically add/subtract memory to a workload
- Maybe your OS does memory auto-tiering



Goal

Demonstrate the value of far memory

- The value of far memory lies in the user, and the workload
  - That being said, CPU core counts keep going up and memory channels don't, all while serialized buses like PCIe continue doubling performance
- There is a lot of work already that shows the utility of far memory
  - But we already talked about replication and bad out of box experiences
- Let's showcase popular, valuable workloads for far memory in an open, repeatable way



Goal

Make it simple and consumable

- Make it simple
  - I shouldn't have to intimately know how the benchmark runs
- Make it digestible
  - I should be able to run it after reading a brief readme.
- Make it clear
  - When this finishes, the report is going to talk about x, y, z

```
[
  {
    "name": "redis-default",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16, "batch_size": 1024 }
  },
  {
    "name": "redis-m-16-ef-128",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 16, "EF_CONSTRUCTION": 128 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  },
  {
    "name": "redis-m-32-ef-128",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 32, "EF_CONSTRUCTION": 128 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  },
  {
    "name": "redis-m-32-ef-256",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 32, "EF_CONSTRUCTION": 256 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  },
  {
    "name": "redis-m-32-ef-512",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 32, "EF_CONSTRUCTION": 512 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  },
  {
    "name": "redis-m-64-ef-256",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 64, "EF_CONSTRUCTION": 256 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  },
  {
    "name": "redis-m-64-ef-512",
    "engine": "redis",
    "connection_params": {},
    "collection_params": {
      "hnsw_config": { "M": 64, "EF_CONSTRUCTION": 512 }
    },
    "search_params": [
      { "parallel": 1, "config": { "EF": 64 } }, { "parallel": 1, "config": { "EF": 128 } }, { "parallel": 1, "config": { "EF": 256 } }, { "parallel": 1, "config": { "EF": 512 } },
      { "parallel": 100, "config": { "EF": 64 } }, { "parallel": 100, "config": { "EF": 128 } }, { "parallel": 100, "config": { "EF": 256 } }, { "parallel": 100, "config": { "EF": 512 } }
    ],
    "upload_params": { "parallel": 16 }
  }
]
```

Just *one* of the many example configs for a popular vectorDB

Goal

Make it simple and consumable

- Make it simple
  - I shouldn't have to intimately know how the benchmark runs
- Make it digestible
  - I should be able to run it after reading a brief readme.
- Make it clear
  - When this finishes, the report is going to talk about x, y, z

Just *one* of the many example configs for a popular vectorDB

Goal

Make it repeatable and comparable-ish

- Lock as much software stack variability as possible
  - Maybe use containers?
- Profile the SUT as much as possible
  - Collect all the system info hardware and software
  - Run basic speeds and feeds test to book end performance before running the benchmarks so you have a yardstick
- Settle on outputs that are transportable across applications, relate output to real world utility
  - Hard, this is the industry discussion. When are we comparing apples to chairs and how do we at least compare fruit?

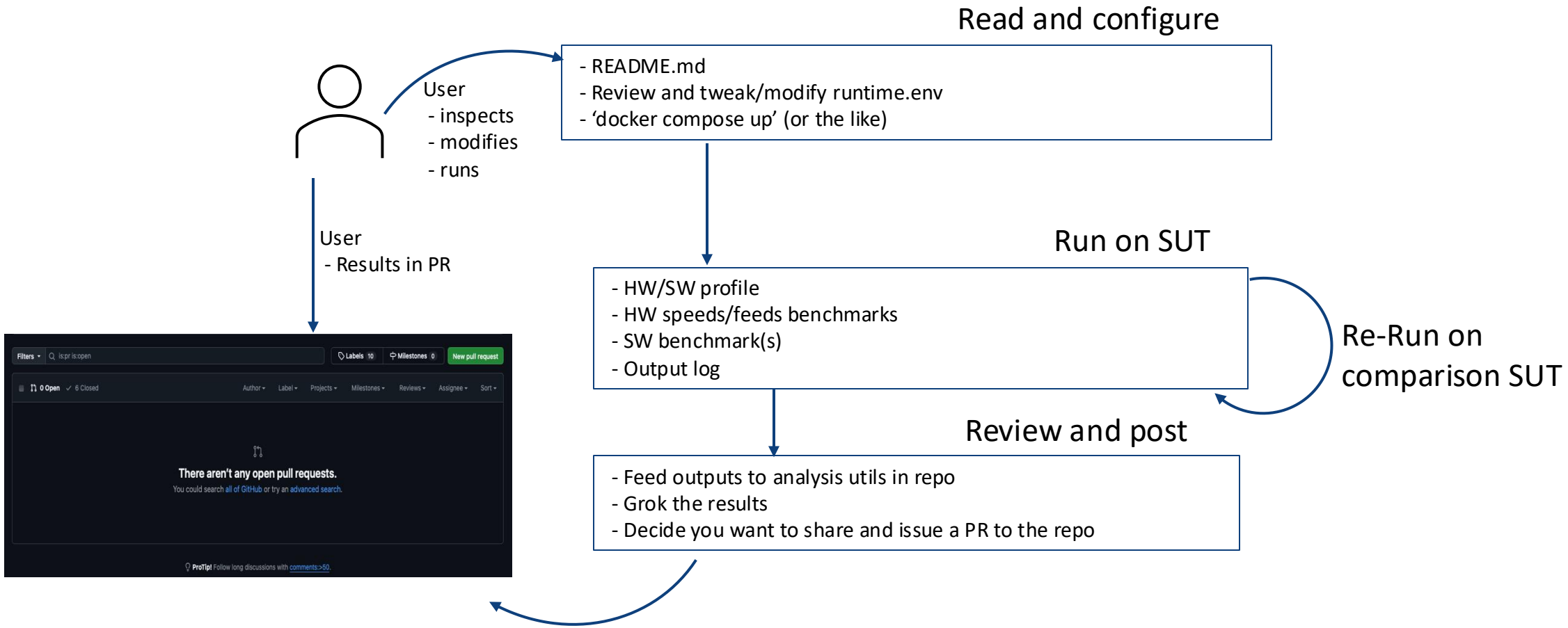
Proposal

“Hermetically sealed containers”…what is that?

- **Hermetically sealed** (adjective) /hɚˌmeṭ.ɪ.kəl.i ˈsiːld/: *separated and protected from very different conditions outside. A container or space tightly closed that no air can leave or enter it.*
  - Put as much of the software stack into containers as possible
- Lock most of the configuration of runtimes into the containers
  - Provide *some* configuration knobs for end users
- Open-source and transparent runtime, but don’t mess with the internals, use the knobs we provide.

Proposal

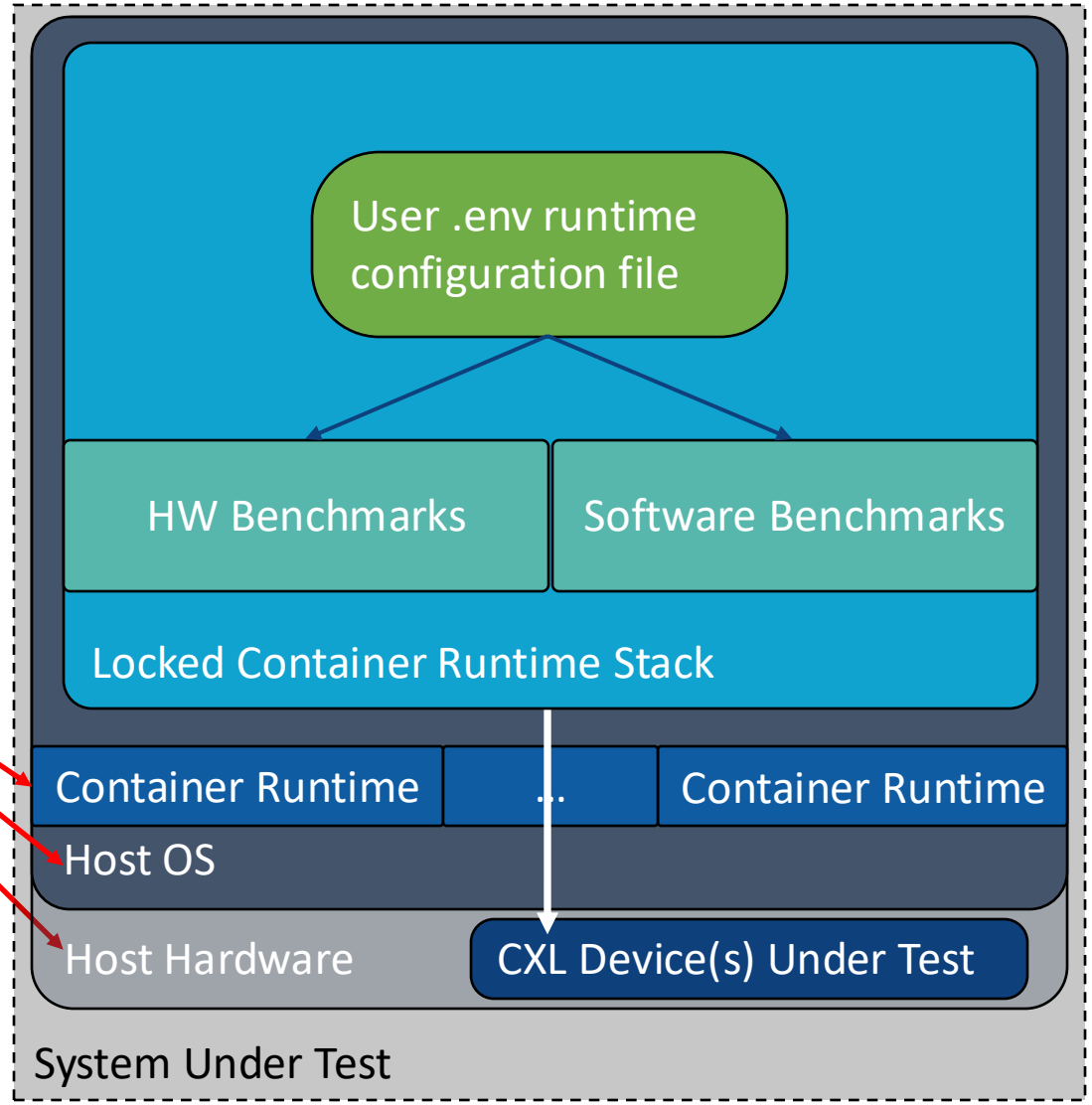
“Hermetically sealed containers”…what is that?



Proposal

“Hermetically sealed containers”…what is that?

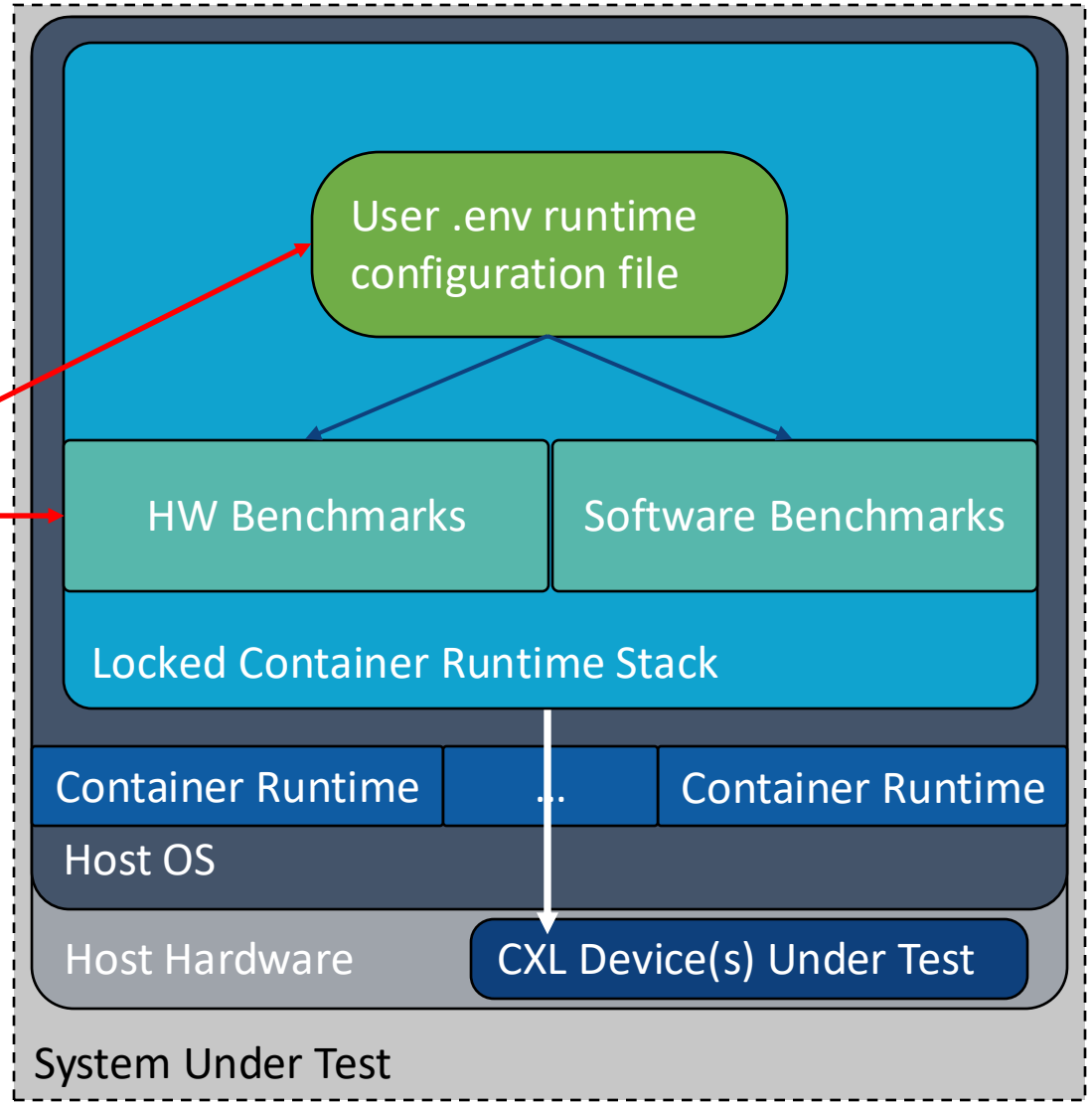
- Profile the system under test both out and in the container
  - Numactl -H
  - Cat /proc/cpuinfo, /proc/meminfo
  - uname –a, lsb_release –a
  - lshw, dmidecode, lspci -vvv
  - [container runtime] --version



Proposal

“Hermetically sealed containers”…what is that?

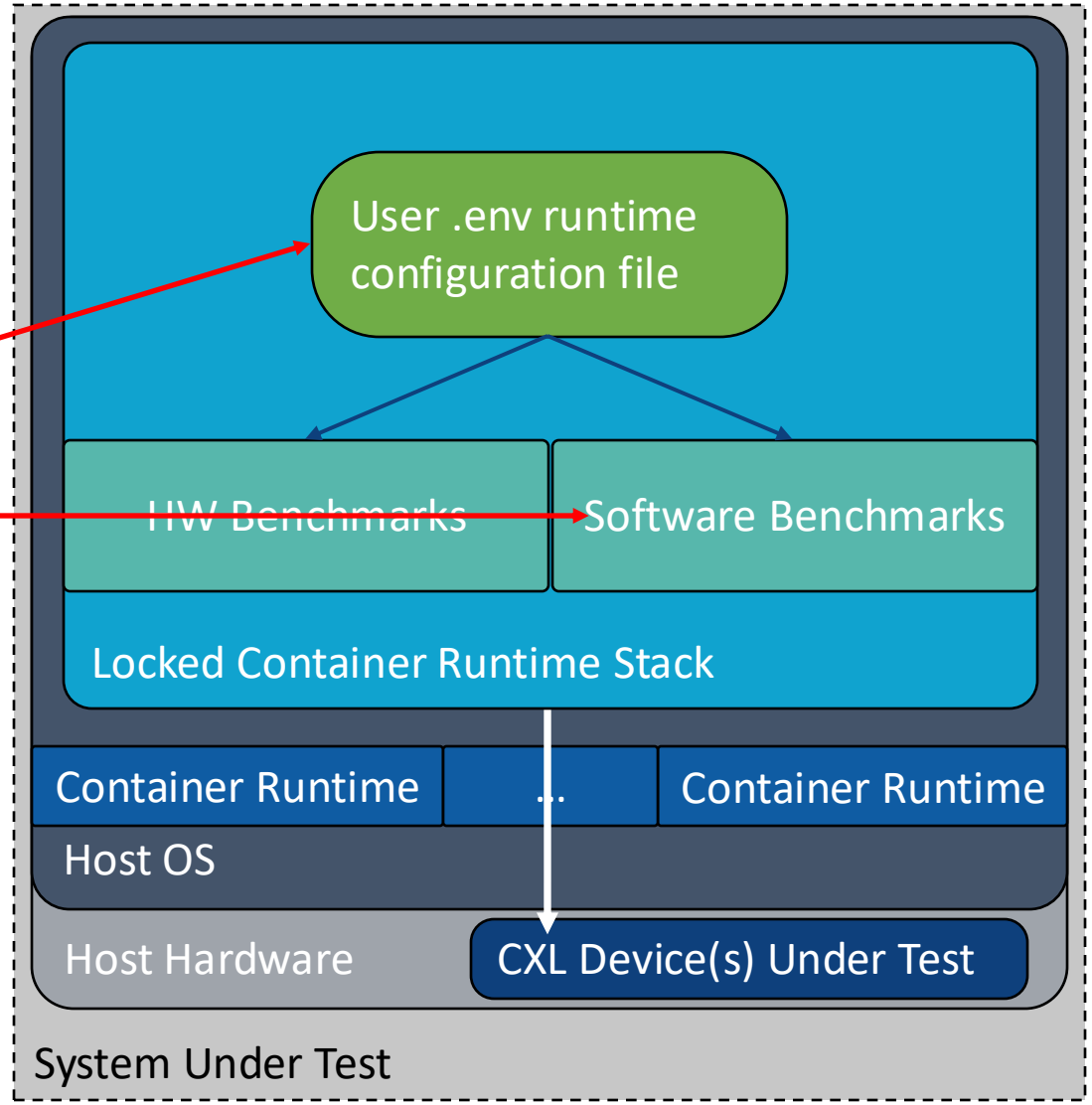
- Measure speeds and feeds of the devices under test



Proposal

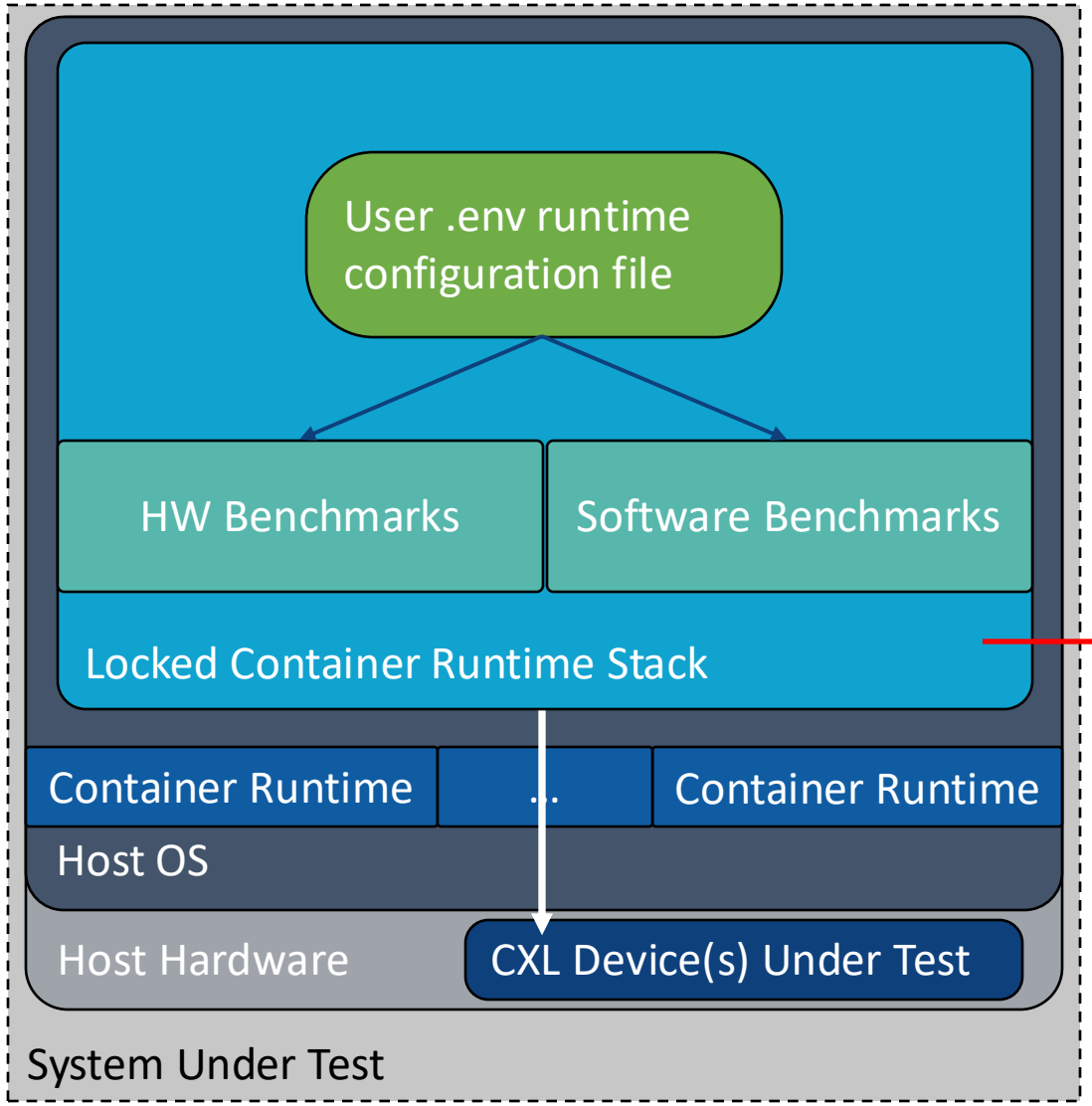
“Hermetically sealed containers”…what is that?

- Run useful™ application benchmarks

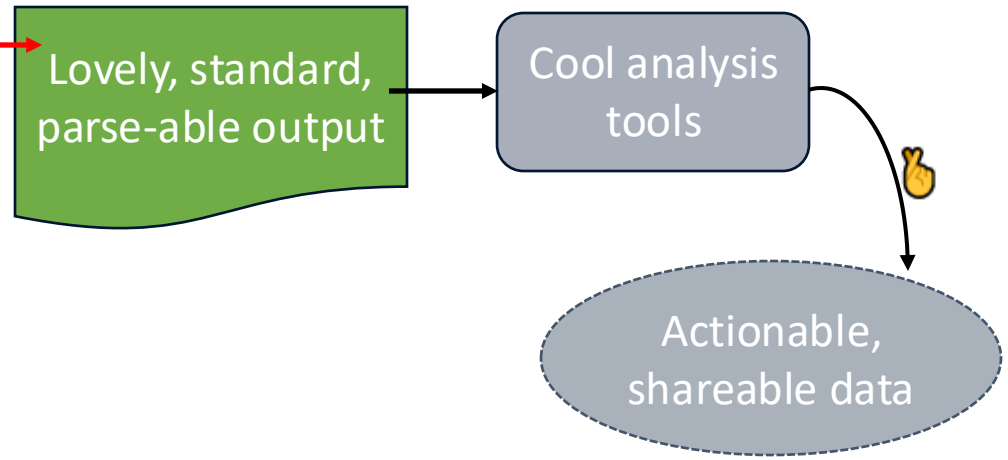


Proposal

"Hermetically sealed containers"…what is that?



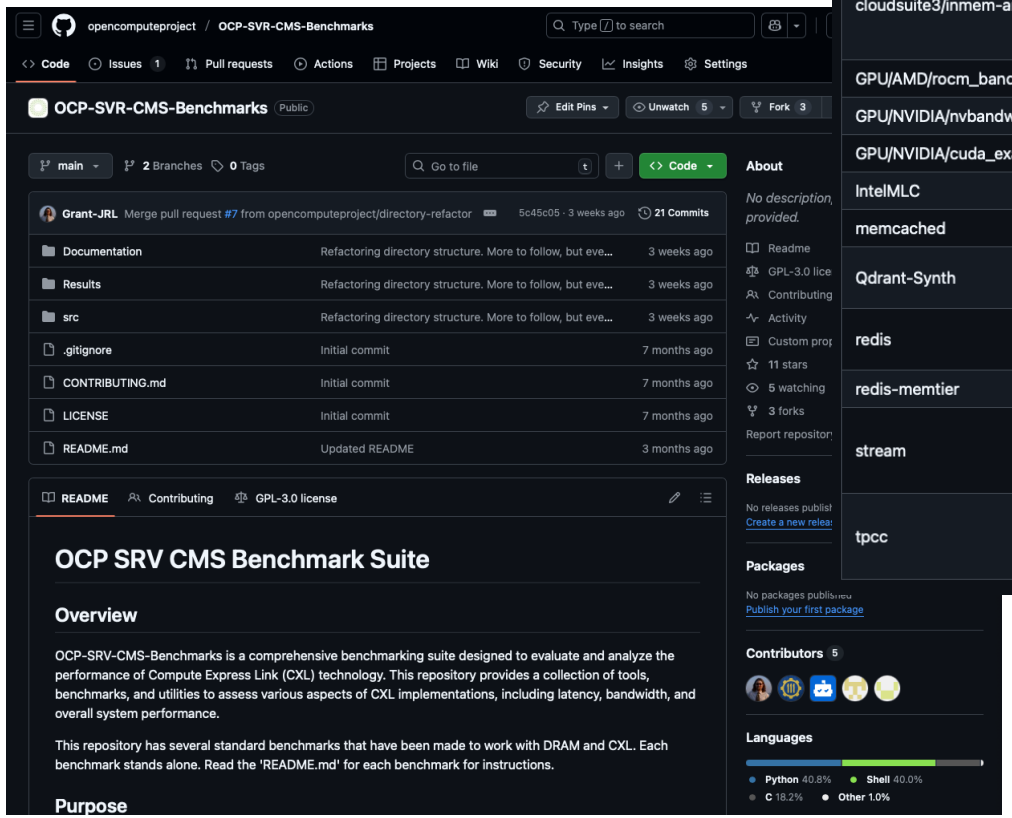
- Return digestible, standardized, output that facilitates replication of results



Proposal

Ecosystem Involvement

- Popular workloads
- Hosting anonymous (or not) results data
- Very interested in collaboration
  - Send your PRs!



Included Benchmarks

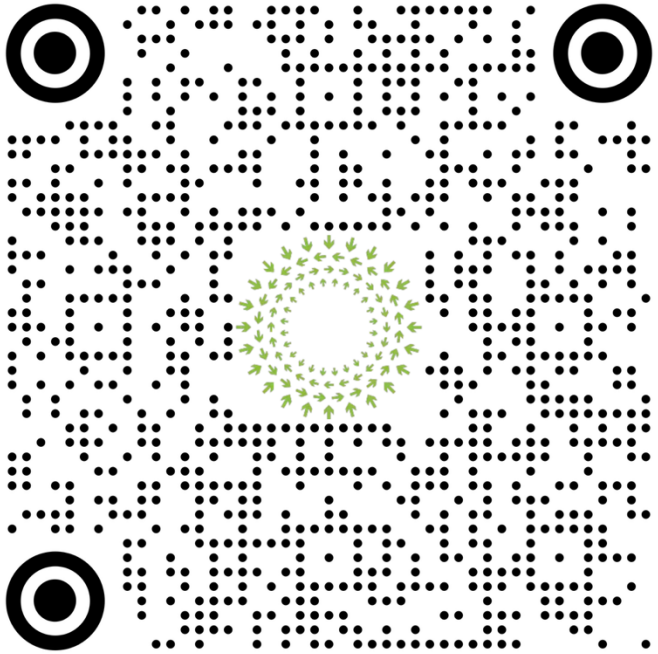
This table shows the list of benchmarks included in this suite:

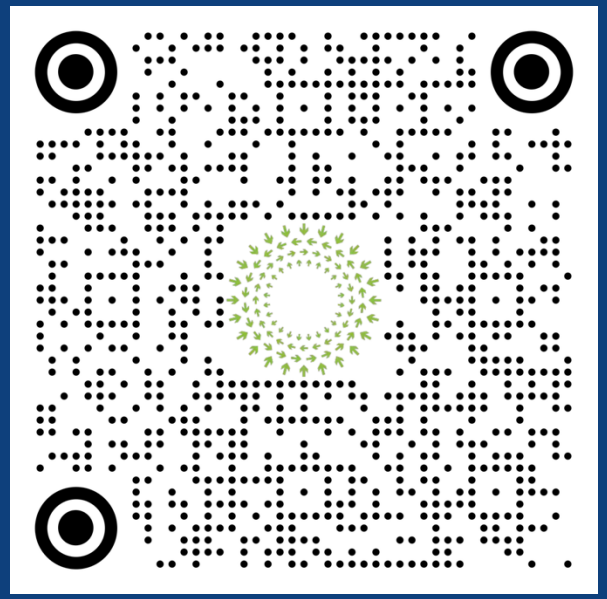
Benchmark	Description
cloudsuite3/graph-analytics	The Graph Analytics benchmark relies on the Spark framework to perform graph analytics on large-scale datasets
cloudsuite3/inmem-analytics	This benchmark uses Apache Spark and runs a collaborative filtering algorithm (alternating least squares, ALS) provided by Spark MLlib in memory on a dataset of user-movie ratings. The metric of interest is the time in seconds for computing movie recommendations.
GPU/AMD/rocm_bandwidth_test	EA tool for bandwidth measurements on NVIDIA GPUs
GPU/NVIDIA/nvbandwidth	EA tool for bandwidth measurements on NVIDIA GPUs
GPU/NVIDIA/cuda_examples	Evaluates the data transfer rates for NVidia GPUs
IntelMLC	Runs the Intel Memory Latency Checker (MLC)
memcached	Memcached is a general-purpose distributed memory-caching system
Qdrant-Synth	Creates synthetic vectors and benchmarks a Qdrant Vector Database running in a Docker Container
redis	Redis is a source-available, in-memory storage, used as a distributed, in-memory key–value database, cache and message broker
redis-memtier	Run the memtier benchmark against a redis instance
stream	The STREAM benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels.
tpcc	TPC-C (Transaction Processing Performance Council Benchmark C), is a benchmark used to compare the performance of online transaction processing systems.

Take-aways

CMS Bench: Open, repeatable benchmarking of far memory technology

- There isn’t an industry consensus on the utility of far memory today
- “Show, don’t tell” benchmarks are useful
- Make it easy, make it digestible
- There isn’t a de-facto effort in the wild yet to tackle this challenge
- The OCP Composable Memory Systems group is taking a crack it it





JACKRABBIT LABS

jrlabs.io
blog.jrlabs.io
contact@jrlabs.io, grant@jrlabs.io