HPC Container Conformance

2024 edition (w/ OCI Working Group intro)

Christian Kniep, 2024-02-03

~~Problems~~ Challenges

module load

People are used to ‘module load’ to pick the right software stack at runtime

- module load gromacs:2021.5

```
$ module load gromacs:2021.5
$ which gmx
/software/graviton2/gcc-7.3.1/gromacs-2021.5-FF/bin/gmx
```

With containers a strategy often used is to use a naming scheme

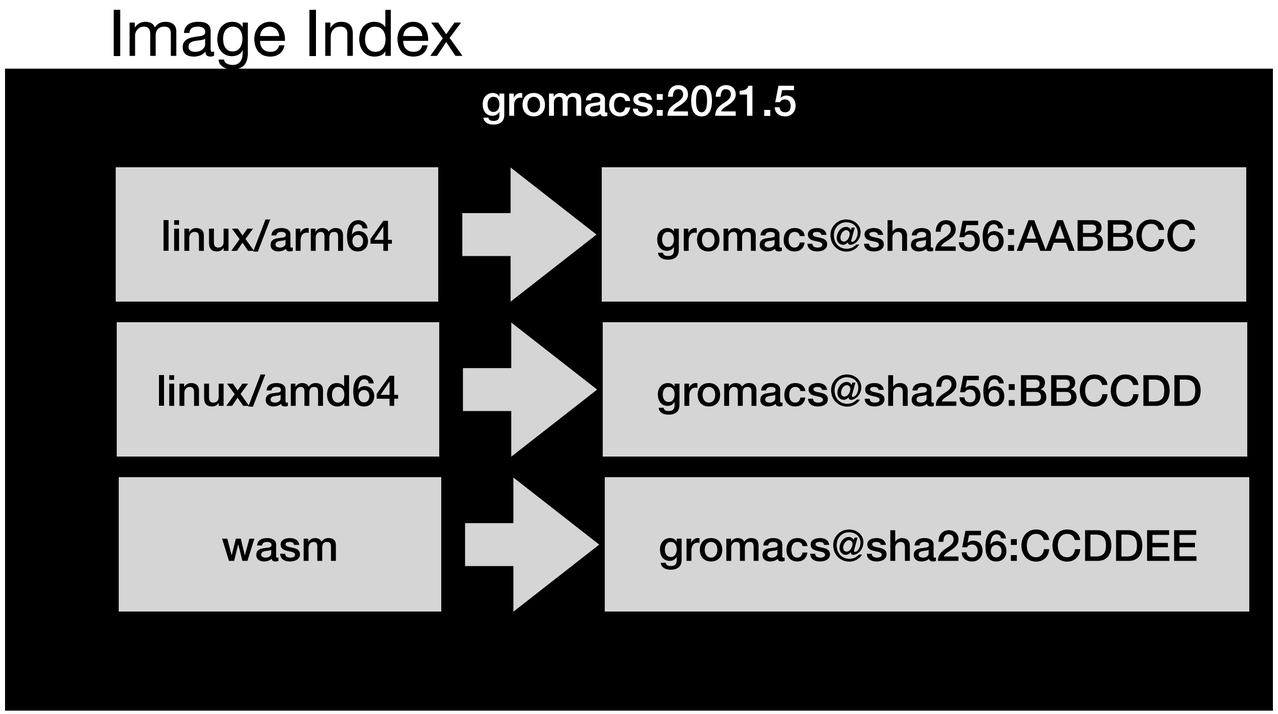
A. cqnib/gromacs:2021.5_graviton2
B. cqnib/gromacs:2021.5_skylake
C. cqnib/gromacs:2021.5_zen3
D. cqnib/gromacs:2021.5_cluster_1

```
$ docker run -ti cqnib/gromacs:2021.5_graviton2
$ which gmx
/opt/view/bin/gmx
```

~~Problems~~ Challenges

Image Index

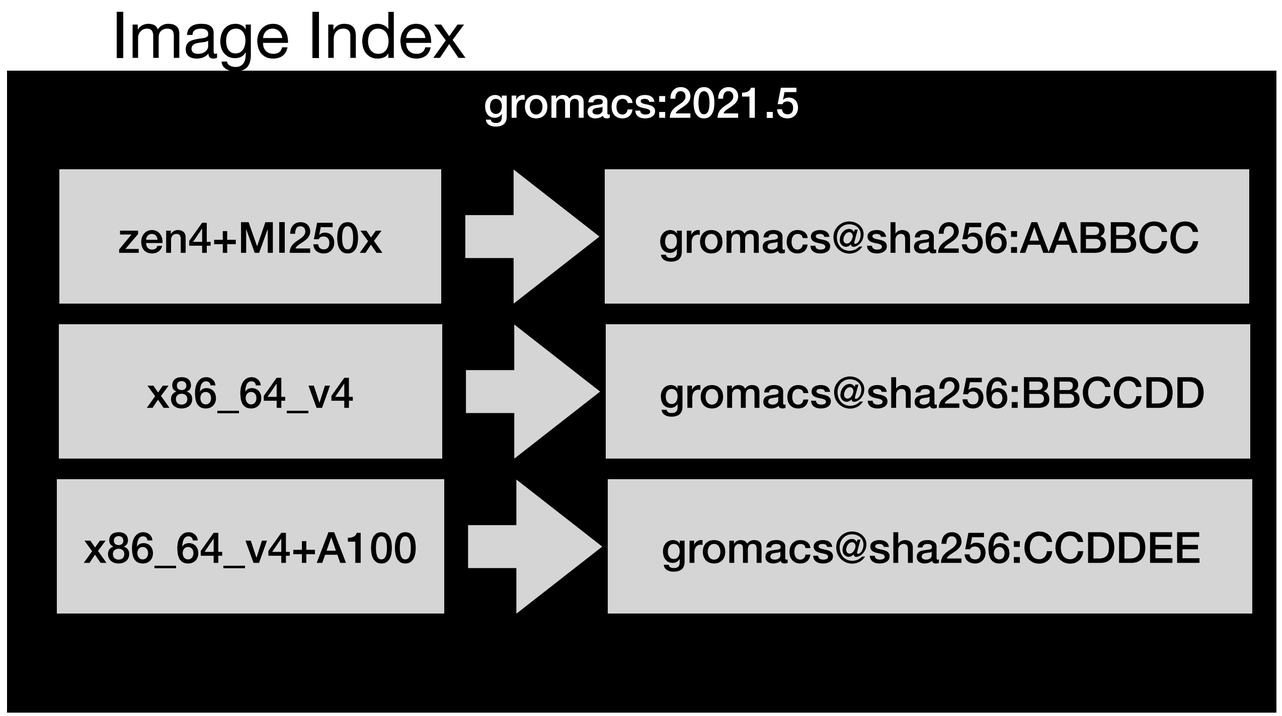
- The platform identifier does not allow for fine grained control
- Runtimes will pick the **first** matching entry, not the best matching (there is no rank)



~~Problems~~ Challenges

Image Index WE DESIRE

- An identifier to specify systems
- Runtimes able to use the identifier to pick the BEST MATCHING system



Early Attempts

Platform (mis-)use

The manifest list uses the (unused) **platform.features** field to define for what the image is optimised for.

```
image: qnib/cv-tf:1.12.0-rev9
manifests:
    image: qnib/cv-tf-dev:1.12.0-rev11
    platform:
      architecture: amd64
      os: linux
    image: qnib/cv-tf-dev:skylake_1.12.0-rev6
    platform:
      features:
        - skylake
    image: qnib/cv-nccl90-tf-dev:1.12.0-rev1
    platform:
      features:
        - nvidia-390-30
    image: qnib/cv-nccl92-tf-dev:1.12.0-rev11
    platform:
      features:
        - nvidia-396-44
    image: qnib/cv-nccl90-tf-dev:b
    platform:
      features:
        - broadwell
        - nvidia-390-30
```

```
platform:
  features:
    - broadwell
    - nvidia-390-30
```

https://qnib.org/blog/2019/02/14/match-node-specific-needs-using-manifest-lists

Early Attempts

Docker Tweak

Each engine is aware of what images are best.

One possible idea is to put it in the `daemon.json`, like this:

```
$ sudo cat /etc/docker/daemon.json
{
  "debug": true,
  "tls": true,
  "tlscacert": "/etc/docker/ca.pem",
  "tlscert": "/etc/docker/cert.pem",
  "tlskey": "/etc/docker/key.pem",
  "tlsverify": true,
  "experimental": true,
  "platform-features": [
    "broadwell",
    "nv-compute-5-2",
    "nvidia-396-44"
  ]
}
```


```
"platform-features": [
  "broadwell",
  "nv-compute-5-2",
  "nvidia-396-44"
]
```


Doing so, the engine will add the `platform-features` automatically, quite like the manual download using:

`--platform=linux/amd64:broadwell:nv-compute-5-2:nvidia-396-44`.

https://qnib.org/blog/2019/02/14/match-node-specific-needs-using-manifest-lists

HPC Container Conformance
(HPC3)

Expected Image Behaviour

Login vs App Container

Expected Image Behaviour

Login Container vs. Application Container

```
$ alias goreleaser="docker run -ti goreleaser/goreleaser"
$ goreleaser
GoReleaser is a release automation tool for Go projects.
Its goal is to simplify the build, release and publish steps while providing variant
customization options for all steps.
```

The above container uses the ENTRYPOINT to start the application in question. All arguments (CMDs) are arguments for the application itself.

Great for application aliases; but for HPC it hinders how the image can be used. Do I need to specify the application (say gmx for GROMACS) or do I start with the arguments?

Expected Image Behavior

Login Container vs. Application Container

For HPC containers we expect to be dropped into a shell (most likely bash)

```
docker run -ti -v $(pwd):/data quay.io/cqnib/gromacs-2021.5_gcc-7.3.1:aarch64
bash-4.2#
```

The look and feel should be similar to logging into a compute node. The environment is prepared to have the application already at your fingertips.

```
$ docker run -ti -v $(pwd):/data -w /data \
     quay.io/cqnib/gromacs-2021.5_gcc-7.3.1:aarch64 gmx  mdrun -s benchRIB.tpr -resethway
                    :-) GROMACS - gmx mdrun, 2021.5-spack (-:
Using 1 MPI thread
Using 8 OpenMP threads
```

ENTRYPOINT	CMD
/bin/bash —rcfile /etc/profile -l -c $* --	/bin/bash

USER within Container

Expected Image Behaviour

USER within Container

The container is going to spawn a process under the UID:GID of an beforehand unknown user. This implies the following:

- Make sure that scripts within the container do not use `whoami` or anything that needs a ‘real’ username
- Make sure the container is able to run as `nobody`

Annotations

What are annotations and labels?

<u>Labels</u> should be considered ground truth for all key/v[redacted]irs

I talked about that last year….

OCI WG: Image Compatibility

OCI Working Groups

WHAT?

The Open Container Initiative (OCI) maintains the key container specifications.

1. **Image Format Specification:** Developing standards for container image formats. This includes the layout, contents, and construction of container images.

2. **Runtime Specification:** Focusing on standardising the runtime environment for containers. This involves how a container should be run and what features it must support.

3. **Distribution Specification:** Concentrating on the mechanisms for distributing container images.

4. **Security:** Working on security aspects of container technology, which can include addressing vulnerabilities, ensuring image provenance, and securing container contents and communication

OCI Working Groups

Image Compatibility

The WG “Image Compatibility” was formed in 2023 and our goal is to create a standard to

- Select an image from a manifest list based on expected performance / compatibility. E.g. pick an image with optimised binaries/libraries.
- Define a OCI wide way of expressing
  - what an image was build for
  - What it expects from the host (needs a kernel module)
  - …

OCI Working Groups

A better way!

Compared to an HPC-only approach we are able to balance this with other groups needs:

- **Runtime questions**: WASM is a thing - and we have questions in common when it comes to picking a runtime over another.
- **Scheduling/Registry**: HPC is great but Container tech is wide spread and humming. The OCI WG makes sure we are aligned with important schedulers (read Kubernetes) and the registry community.
- **Process**: OCI WG are oiled machines in standardisation

Where are we now?

OCI State of Affairs

What did we do so far

1. Discussions around use-cases
2. Brainstorming of implementations (as we discuss use-cases)

Use-case #1

The Image Author

1. wants to create an image compatibility definition
2. most likely wants not to do it manually (@vosch wrote a tool [1]), w/ functionality ideally included in build pipeline (EasyBuild, Spack)
3. …

[1] github.com/supercontainers/compspec-go

Use-case #2

The System Admin

1. wants to check whether an image is going to run w/o downloading it first
2. wants to collect information which images of which kind are run
3. wants to validate that all (the important) applications are going to work on a new system before switching
4. wants to be able to dig deeper into the comp-spec to understand how to setup a system to leverage an image to the fullest.

Use-case #3

The End User

1. just wants it to just work :) (SysAdmin and image author should figure it out)

Use-case #4-9

4. **Domain Architect:** person that provides the expertise for how images should be described in the compatibility specification
5. **Tool Writer**: person that builds tools to manage images on registries and other locations.
6. **Deployment Engineer**: Person who owns managing the application deployment
7. **Registry Maintainer**: person maintaining the registry (e.g., Docker Hub, Harbor)
8. **OCI Specification Maintainer:** someone from the open containers initiative (aka us)
9. **Security Administrator:** Person responsible for securing the environment.

Links

Resources to follow up on

- https://github.com/supercontainers/compspec-go
  An extractor tool Vanessa Sochat
- github.com/opencontainers/tob/blob/main/proposals/wg-image-compatibility.md
  Working Group proposal to the OCI
- github.com/kubernetes-sigs/node-feature-discovery
  Node feature discovery (NFD) for Kubernetes

hpc-containers
hpc-containers.slack.com

Open Container Initiative
opencontainers.slack.com

ISC'24: HighPerf Container Workshop

5/16 9AM - 6PM

ISC'24: Friends of Container Boat Trip

5/13 7PM (-ish, keep an eye on that)

container-in-hpc.org/isc/2024/overview/index.html