Part 1.2 HPC ready container images

Jorge Ejarque (BSC)
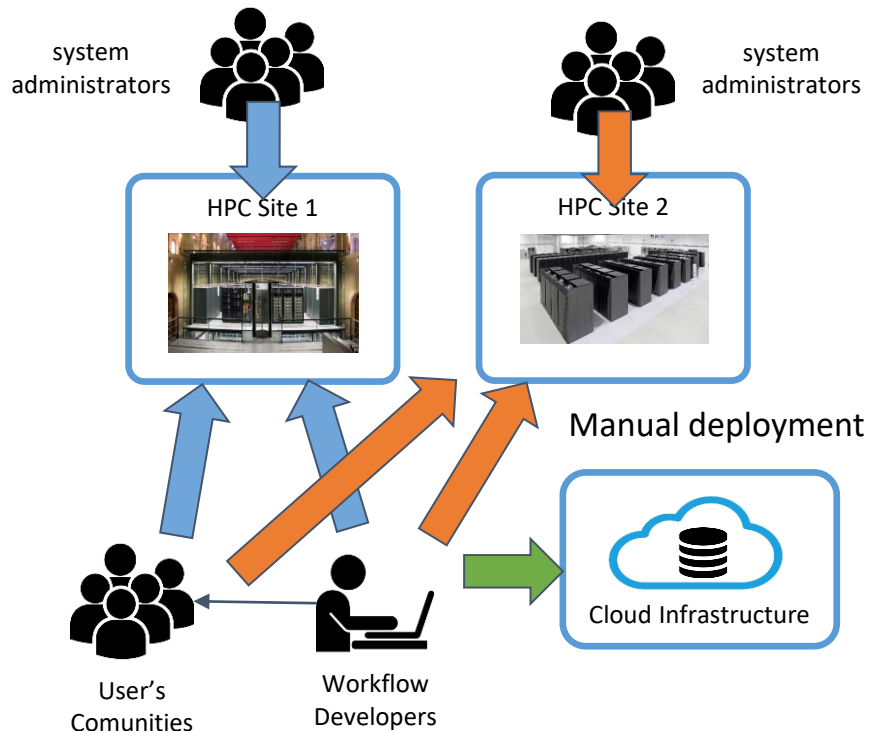
ISC 23 – Hamburg 21st of May 2023
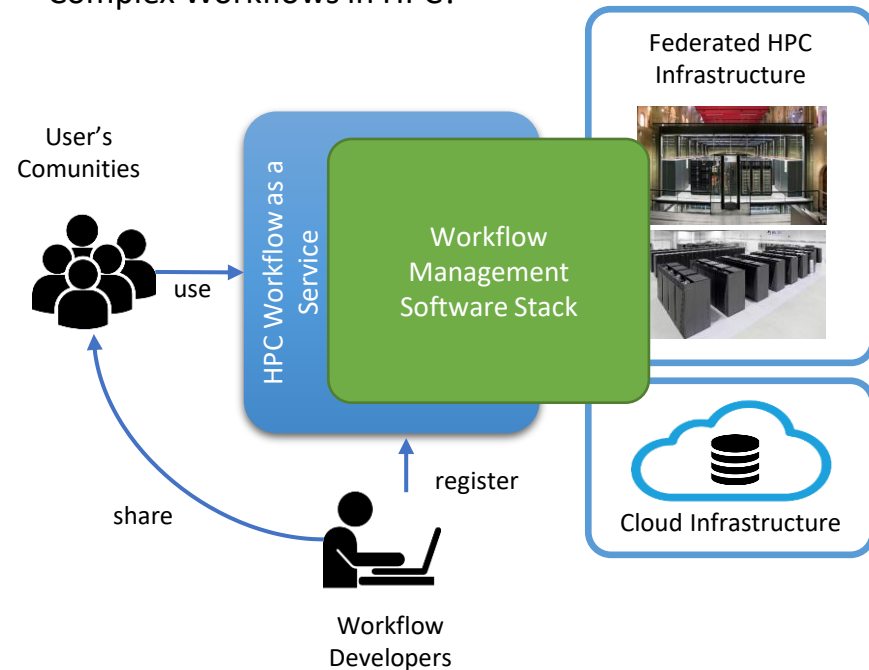
Deployment in HPC Environments

eFlows4HPC

Current approach

system administrators

system administrators

HPC Site 1

HPC Site 2

Manual deployment

Cloud Infrastructure

User's Comunities

Workflow Developers

Can we apply something like FaaS for Complex Workflows in HPC?

User's Comunities

HPC Workflow as a Service

Workflow Management Software Stack

Federated HPC Infrastructure

Cloud Infrastructure

use

share

register

Workflow Developers

FaaS vs. HPCWaaS

Function EP

FaaS Offering

Function Container (OS+runtime)
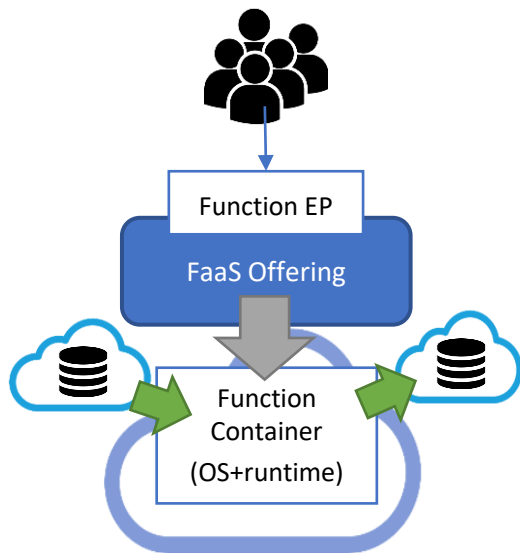
Similarities

- Easy to use for final user
- **Automate deployment & execution**
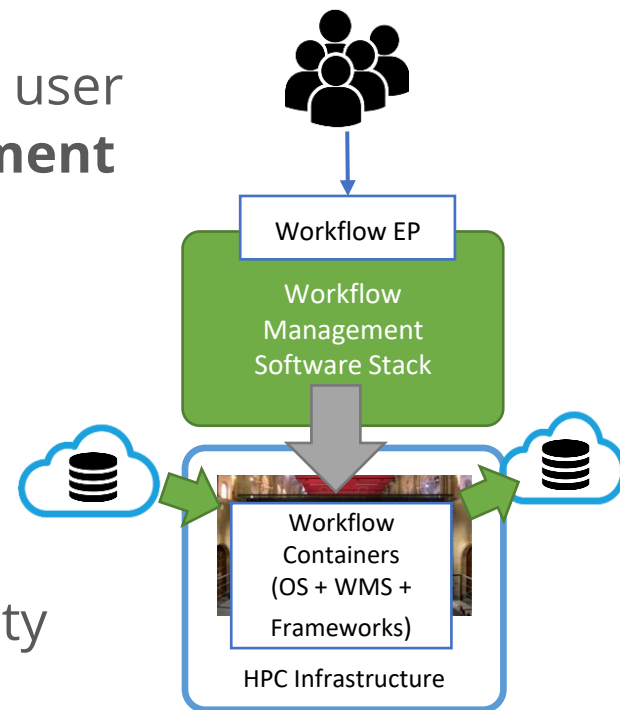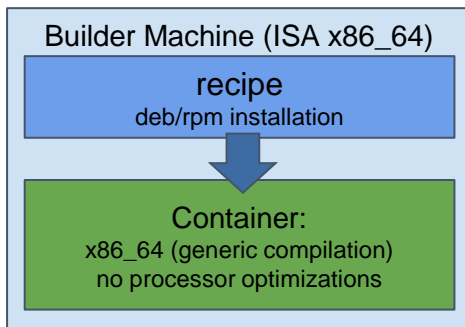- Data integration
- **Containers**

Differences

- Restrictions
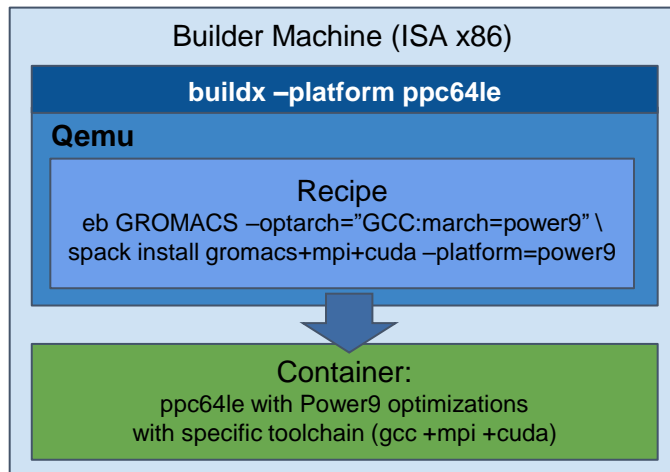- Deployment and Execution Complexity
- **Performance**

Workflow EP

Workflow Management Software Stack

Workflow Containers (OS + WMS + Frameworks)

HPC Infrastructure

Containers and HPC

Standard container image creation

Builder Machine (ISA x86_64)

recipe
deb/rpm installation

Container:
x86_64 (generic compilation)
no processor optimizations

- **Simplicity for deployment**
  - Just pull or download the image
- **Trade-Off performance/portability**
  - Architecture Optimizations
- **Accessing Hardware from Containers**
  - MPI Fabric /GPUs
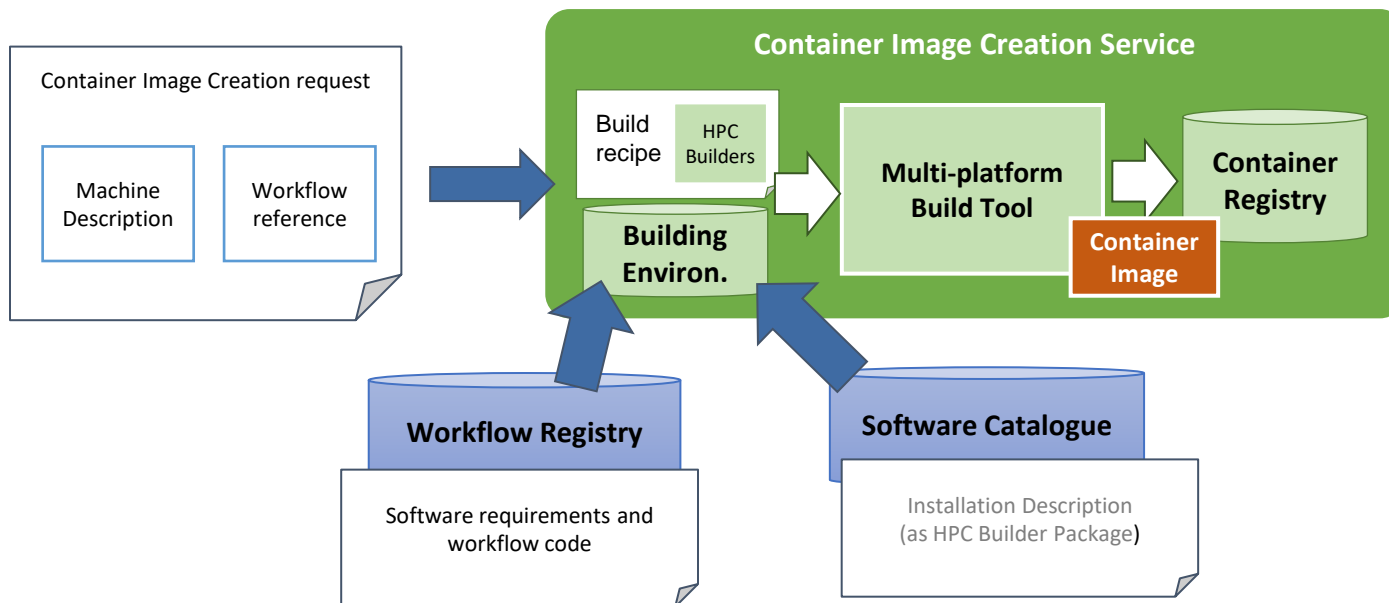- **Host-Container Version Compatibility**

HPC Ready Containers

eFlows4HPC approach

Builder Machine (ISA x86)
buildx –platform ppc64le
Qemu

Recipe
eb GROMACS –optarch="GCC:march=power9" \
spack install gromacs+mpi+cuda –platform=power9

Container:
ppc64le with Power9 optimizations
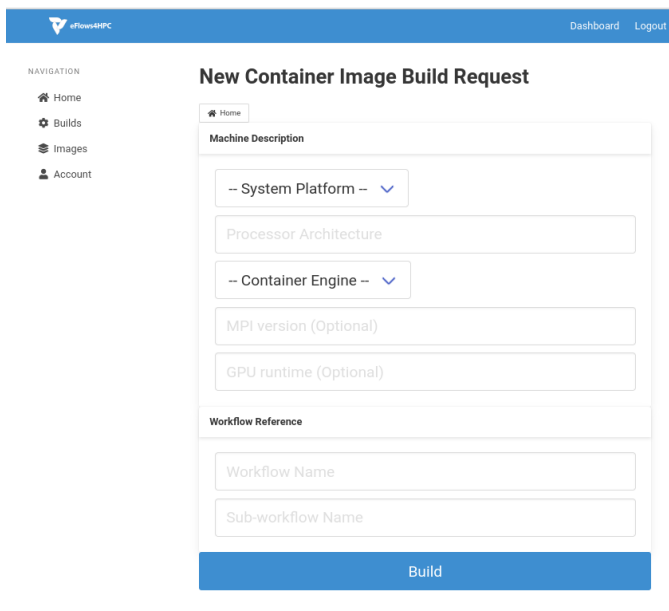with specific toolchain (gcc +mpi +cuda)

- **Methodology to allow the creation containers for specific HPC system**
  - Leverage HPC and Multi-platform container builders
- **It is tight to do by hand but let's automate!**

Container Image Creation Service

eFlows4HPC

Container Image Creation request

| Machine Description | Workflow reference |

Container Image Creation Service

Build recipe — HPC Builders

Building Environ.

Multi-platform Build Tool

Container Image

Container Registry

Workflow Registry

Software requirements and workflow code

Software Catalogue

Installation Description (as HPC Builder Package)

Container Image Creation Service

- ## Web Interface

- ## REST Interface and CLI

Example: Pillar I for MN4

eFlows4HPC

Container Image Creation request

```
"machine": {
    "platform": "linux/amd64",
    "architecture": "skylake_avx512",
    "container_engine": "singularity",
    "mpi" : intel-mpi@2018.4
    },
```

```
"workflow": "rom_pillar_I",
 "step_id": "reduce_order_model"
```

Container Image Creation Service

Building Environ.

Dockerfile   Spack install

| Concretized spack environment | Spack package repo and config. |

Buildx + qemu binfmt (support different platforms)

Container Image

Container Registry

Workflow Registry

Generic Spack environment

```
spack:
    specs:
        - compss
        - py-dislib
        - kratos@9.1.4 apps=LinearSolversApplication, \
                        FluidDynamicsApplication, \
                        StructuralMechanicsApplication, \
                        ConvectionDiffusionApplication
```

Software Catalogue

Spack Packages repo

```
class Compss(Package):
    url = ..
    version('2.10', sha256='...', preferred=True)
    ….
    depends_on('python')
    depends_on('openjdk')
    …
    def install(self, spec, prefix):
        install_script = Executable('./install')
        install_script('-A', '--only-python-3', prefix.compss)

    def setup_run_environment(self, env):
        env.set('COMPSS_HOME', self.prefix.compss)
        env.prepend_path('PATH', self.prefix.compss + '

    def setup_build_environment(self, env):
        ….
```

```
class Kratos(CMakePackage):
    …
    variant('mpi', default=False, description='..')
    depends_on('mpi', when='+mpi')

    def cmake_args(self):
        args = []
        if self.spec.variants['mpi'].value == True:
            args.append('USE_MPI=ON')
        else:
            args.append('USE_MPI=OFF')
        return args
    …
```

```
class PyDislib(PythonPackage):
    …
    depends_on('python@3:', type=('build', 'run'))
    depends_on('py-scikit-learn@0.23^pyscipy@1.5,type=('run'))
```
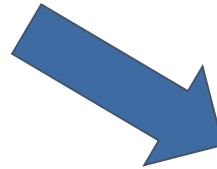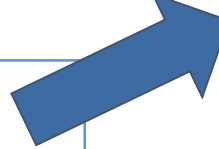
8

Spack Environment Concretization

```
"machine": {
    "platform": "linux/amd64",
    "architecture": "skylake_avx512",
    "container_engine": "singularity",
    "mpi" : intel-mpi@2018.4
    },
```

```
spack:
    specs:
        - compss
        - py-dislib
        - kratos@9.1.4 apps=LinearSolversApplication, \
                        FluidDynamicsApplication, \
                        StructuralMechanicsApplication, \
                        ConvectionDiffusionApplication
```

```
spack:
    specs:
        - compss
        - py-dislib
        - kratos@9.1.4 apps=LinearSolversApplication, \
                        FluidDynamicsApplication, \
                        StructuralMechanicsApplication, \
                        ConvectionDiffusionApplication
        - intel-mpi@2018.4
    concretization: together
    view: /opt/view
    packages:
        all:
            target: [' skylake_avx512 ']
```
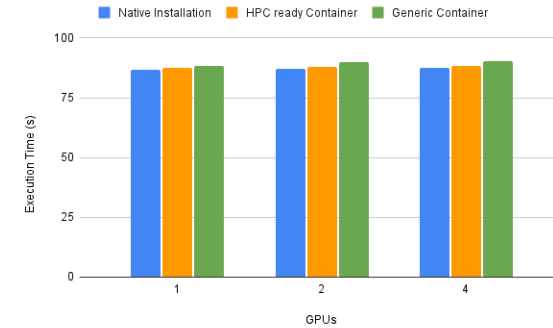
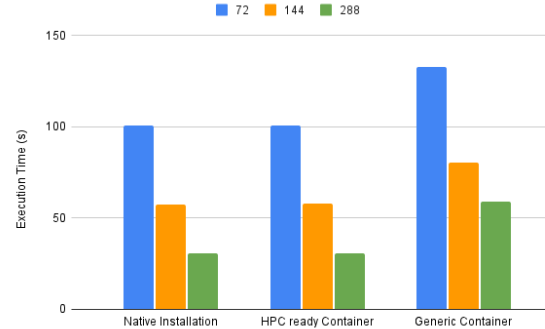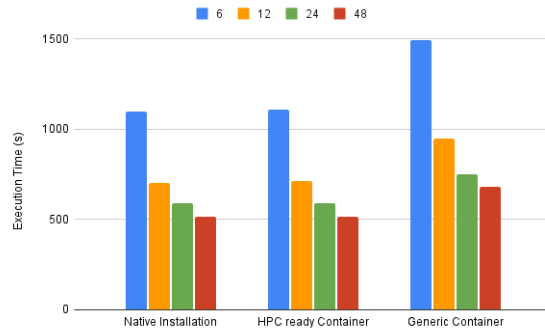# Performance



Kratos Multiphysics (shared memory)
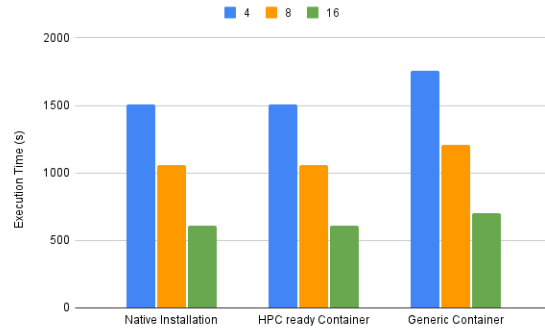
FESOM2 (MPI)
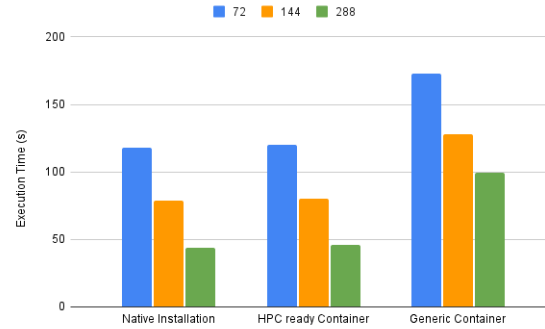
Tsunami-HySEA    CTE-Power

Nord3

MN4

# Take away message

- **HPC ready container images**
  - Advantages:
    - Reduce the deployment complexity
    - Performance close to bare metal
  - Drawbacks:
    - Larger building times compared to OS packages
      - ✓ Require more compilations
      - ✓ Configure spack binary caches
    - Less portable
      - ✓ CPU architecture, mpi, gpu versions