7th EUM

Building a heterogeneous MPI stack with EasyBuild

Alex Domingo

VRIJE UNIVERSITEIT BRUSSEL

VUB HPC

VLAAMS SUPERCOMPUTER CENTRUM

Vlaanderen is supercomputing

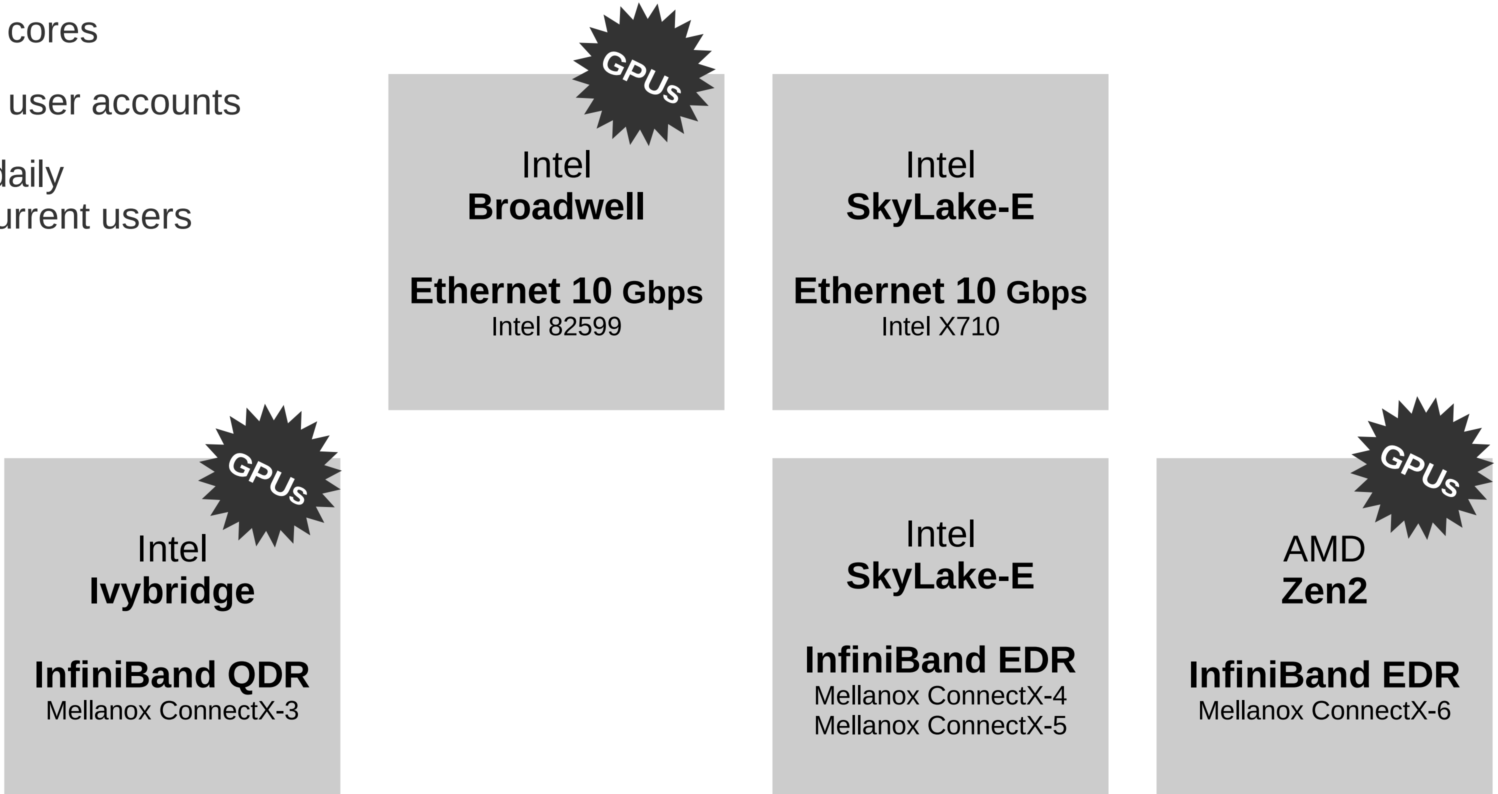
WHO AM I?

Hi! I’m Alex (github: @lexming)

- Background
  - PhD in computational chemistry
  - User of Linux and FOSS in general since mid 2000’s
- **Present time: HPC team of VUB** since 2019 (hpc.vub.be)
  - Horizontal team: Linux sysadmin, software optimization, direct user support, hardware hammering
  - Maintainer of EasyBuild (easybuild.io): open source software build and installation framework for HPC
- Free time: eats chocolate and plays with raspberry pis

VUB TIER-2 HPC (HYDRA)

- 3648 cores
- ~500 user accounts
- ~50 daily concurrent users



GPUs

Intel **Broadwell**

Ethernet 10 Gbps
Intel 82599

Intel **SkyLake-E**

Ethernet 10 Gbps
Intel X710

GPUs

Intel **Ivybridge**

InfiniBand QDR
Mellanox ConnectX-3

Intel **SkyLake-E**

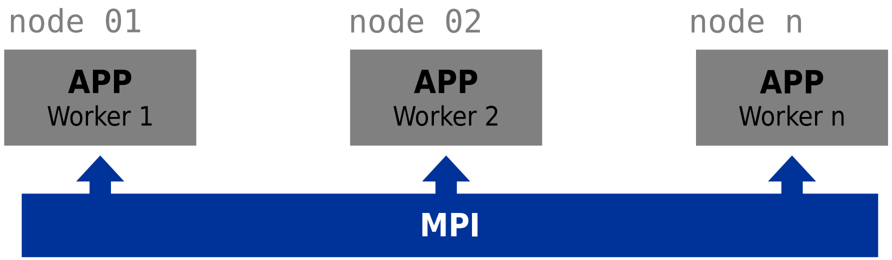
InfiniBand EDR
Mellanox ConnectX-4
Mellanox ConnectX-5

GPUs

AMD **Zen2**

InfiniBand EDR
Mellanox ConnectX-6

DISTRIBUTED PARALLEL COMPUTING



node 01

APP
Worker 1

node 02

APP
Worker 2

node n

APP
Worker n

MPI

Application executing in parallel on multiple CPU cores and multiple nodes

Message Passing Interface (MPI)

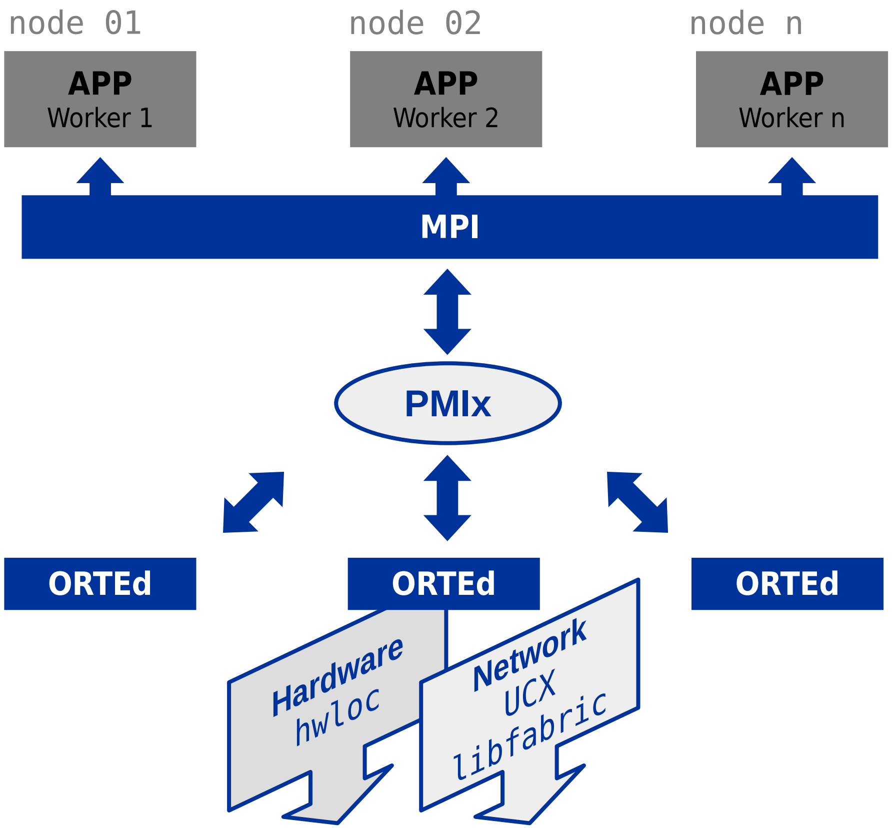
- Communication between application workers

MPI is available in high level toolchains

► foss gompi GCC GCCcore

► intel iimpi intel-comp. GCCcore

OPENMPI



OpenMPI is the main open source MPI implementation

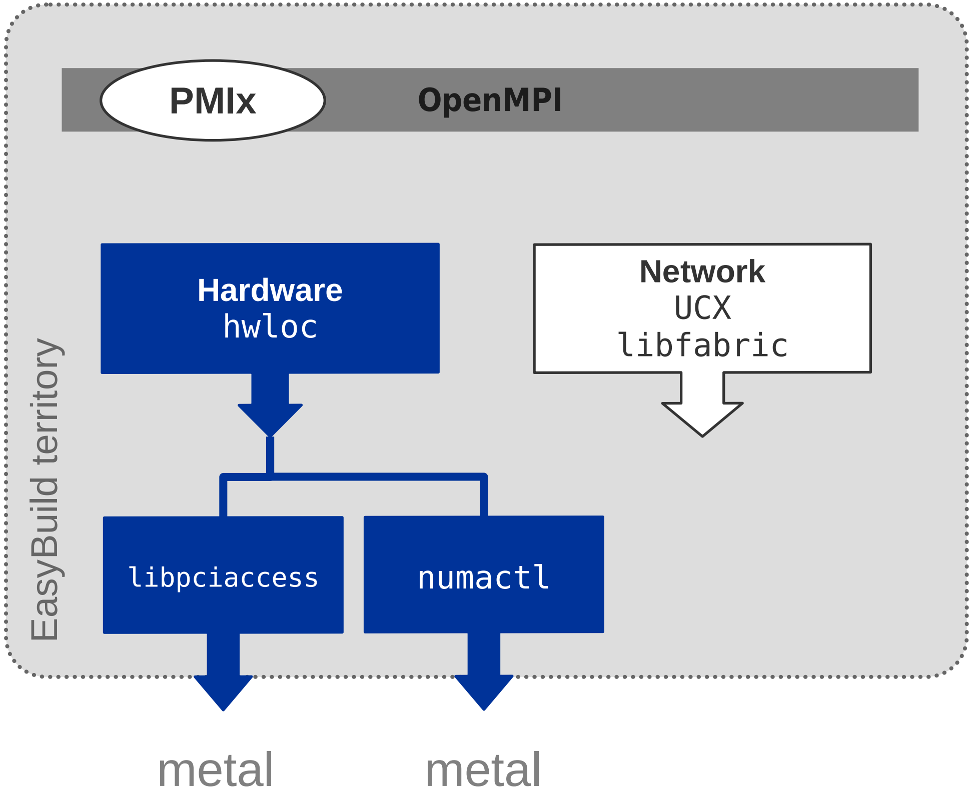
- Not just the MPI API, it has its own runtime environment (ORTE)

```
dependencies = [
    ('zlib', '1.2.11'),
    ('hwloc', '2.5.0'),
    ('libevent', '2.1.12'),
    ('UCX', '1.11.2'),
    ('libfabric', '1.13.2'),
    ('PMIx', '4.1.0'),
]
```

OpenMPI-4.1.1-GCC-11.2.0.eb

OpenMPI
www.open-mpi.org

HWLOC



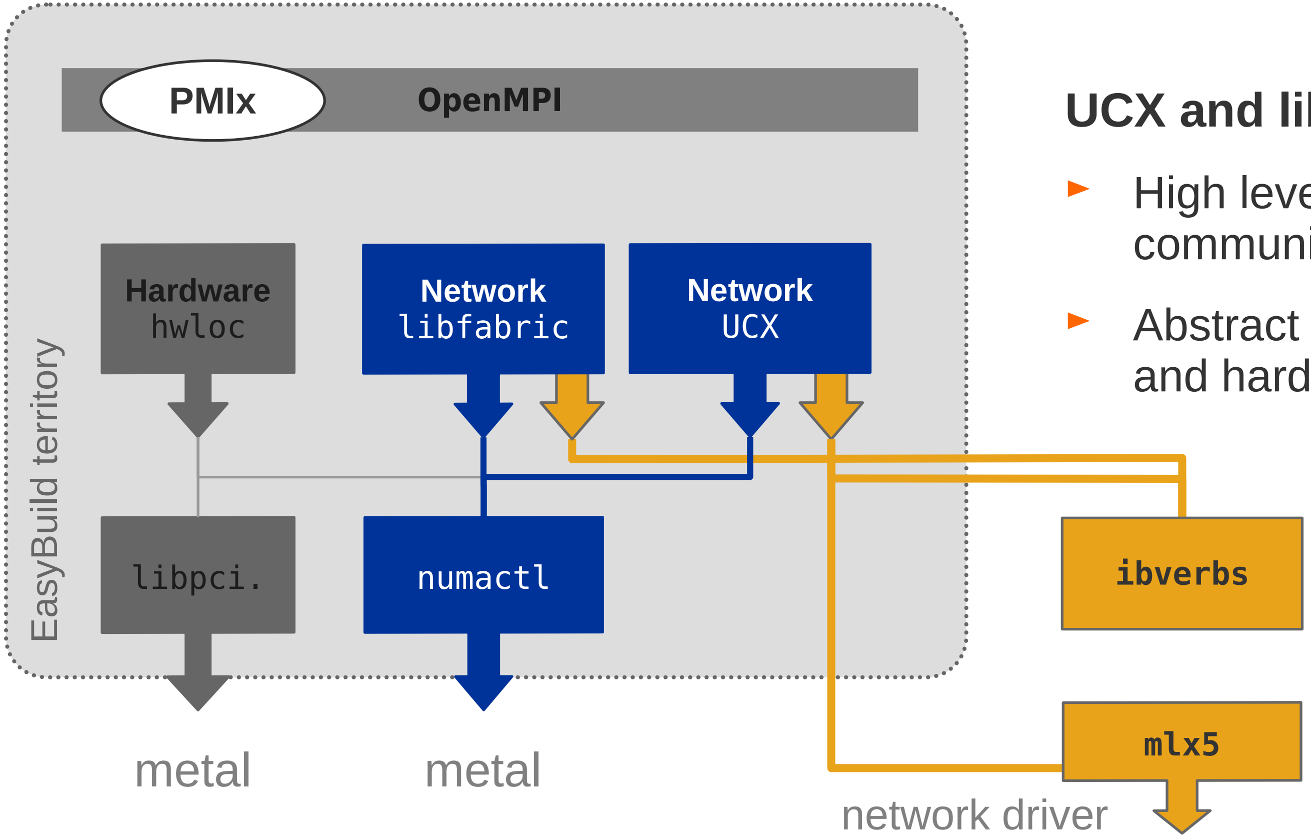
Portable Hardware Locality (hwloc)

- Abstraction of the hardware topology
  - CPU cores
  - Memory
  - NICs
  - GPUs

SAFE

hwloc
www.open-mpi.org/projects/hwloc/

UCX AND LIBFABRIC



UCX and libfabric have similar goals

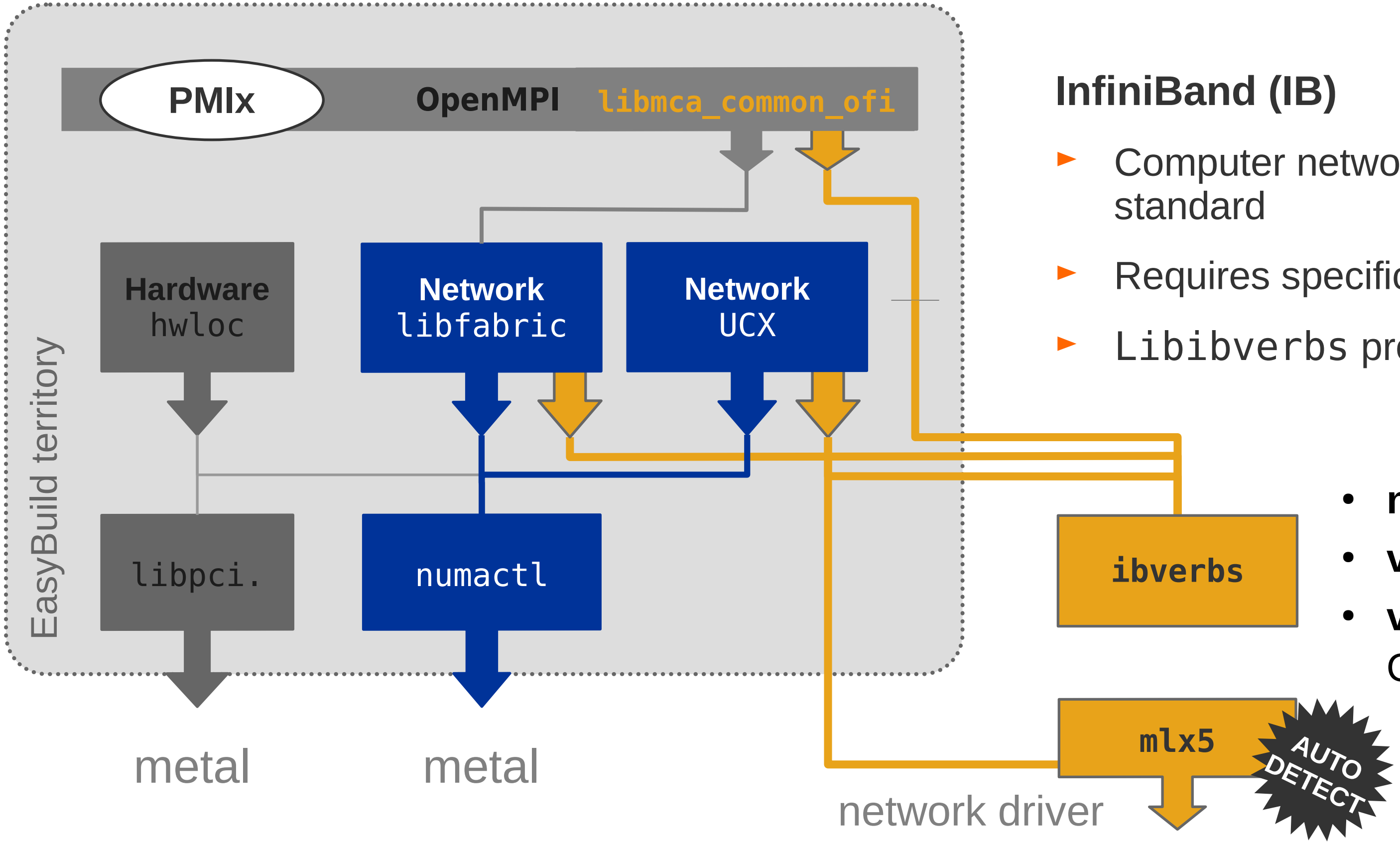
- High level framework for network communications
- Abstract underlying network fabrics and hardware drivers

```
osdependencies =
     [OS_PKG_IBVERBS_DEV]
```

libfabric
libfabric.org

UCX
openucx.org

INFINIBAND



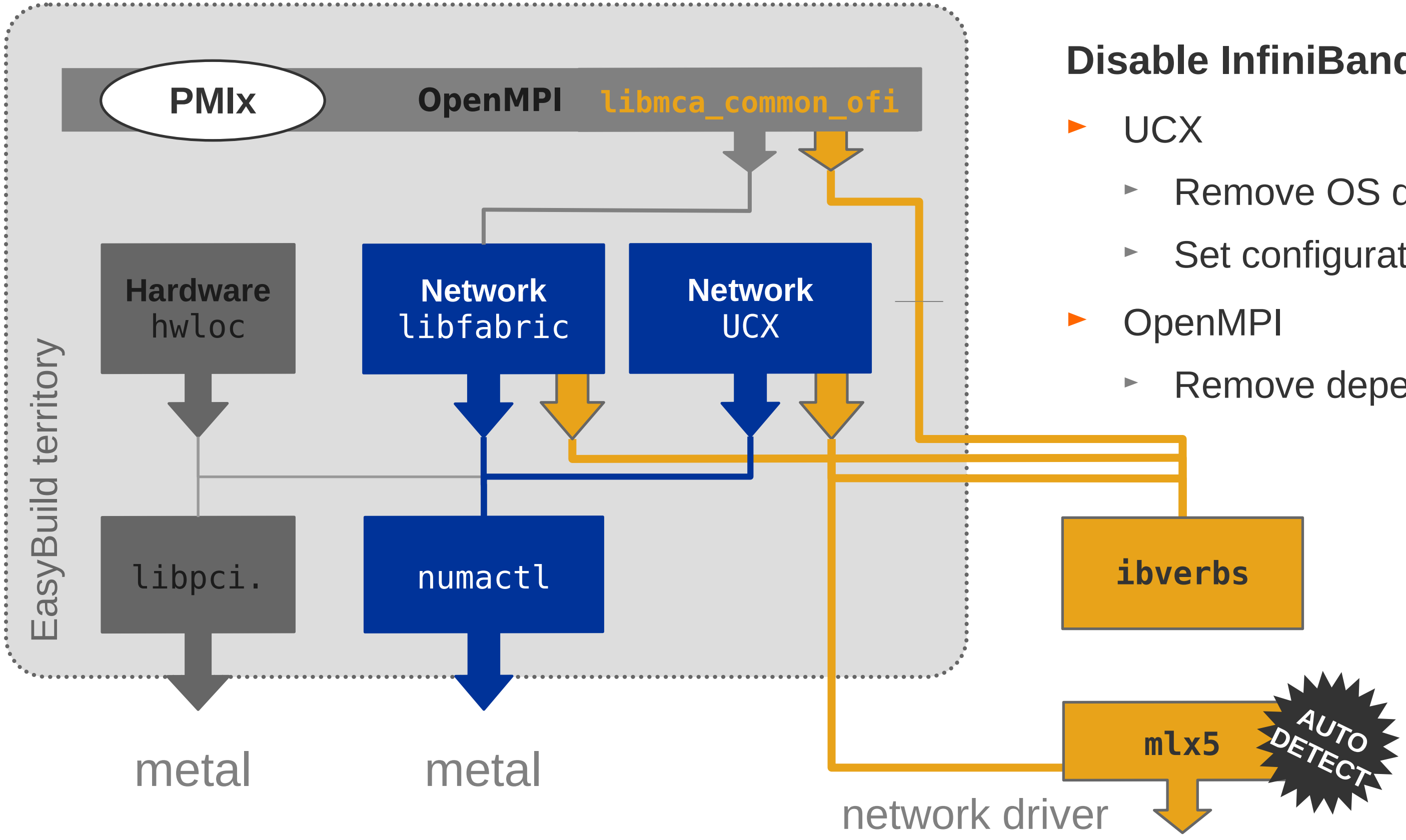
InfiniBand (IB)

- Computer networking communications standard
- Requires specific hardware
- `Libibverbs` provides access to RDMA in IB

- **not installed**
- **v1.5**.22.4 from **CentOS** repos
- **v1.14**.35.0 from **Mellanox** OFED 5.4

RDMA verbs
github.com/linux-rdma/rdma-core

SELECTIVE INFINIBAND SUPPORT



Disable InfiniBand in OpenMPI

- UCX
  - Remove OS dep on `libibverbs`
  - Set configuration options
- OpenMPI
  - Remove dependency on `libfabric`

RDMA verbs
github.com/linux-rdma/rdma-core

EASYBUILD HOOKS

EasyBuild provides hooks for *all** steps executed during the installation

```
$ eb --hooks=/path/to/hooks/code.py
```

You can check the available hooks you can tap into with

```
$ eb --avail-hooks
List of supported hooks (in order of execution):
start_hook
parse_hook
module_write_hook
pre_fetch_hook
post_fetch_hook
[...]
```

EasyBuild Hooks
docs.easybuild.io

EASYBUILD HOOKS

What does a hook look like?

hook.py

```python
def pre_configure_hook(self, *args, **kwargs):
    """Hook at pre-configure level to alter configopts"""

    if self.name == 'OpenMPI':
        self.log.info("[pre-configure hook] Enable XXX")
        self.cfg.update('configopts', "--with-XXX")
```

```
$ eb --hooks=/path/to/hook.py OpenMPI-4.1.1-GCC-11.2.0.eb
```

SELECTIVE INFINIBAND SUPPORT

- Enable/disable support for IB in UCX programmatically with EB hooks

```
IB_OPT_MARK = ['verbs', 'rdma']

def pre_configure_hook(self, *args, **kwargs):
    """Hook at pre-configure level to alter configopts"""

    if self.name == 'UCX':
        ec_config = self.cfg['configopts'].split(' ')
        ib_free_config = [opt for opt in ec_config
                          if not any(mark in opt for mark in IB_OPT_MARK)]

        if {machine has IB}:
            ib_opt = '--with-verbs'  # enable IB
        else:
            ib_opt = '--without-verbs --without-rdmacm'  # disable IB

     ib_config = ib_free_config + [ib_opt]
     self.cfg['configopts'] = ' '.join(ib_config)
```

SELECTIVE INFINIBAND SUPPORT

- Handle OS dependency on `libibverbs` in **UCX** and `libfabric` in **OpenMPI**

```
from easybuild.framework.easyconfig.constants import EASYCONFIG_CONSTANTS

def parse_hook(ec, *args, **kwargs):
    """Alter the parameters of easyconfigs"""

    if {machine does *not* have IB}:
        if ec.name == 'OpenMPI':
            # remove dependency on libfabric in non-IB nodes
            ec['dependencies'] = [d for d in ec['dependencies'] if d[0] != 'libfabric']

        if ec.name == 'UCX':
            # remove any OS dependency on verbs in non-IB nodes
            pkg_ibverbs = EASYCONFIG_CONSTANTS['OS_PKG_IBVERBS_DEV'][0]
            ec['osdependencies'] = [d for d in ec['osdependencies'] if d != pkg_ibverbs]
            ec.log.info("[parse hook] OS dependencies: %s", ec['osdependencies'])
```

IB/NON-IB SOFTWARE MODULES

Software is located in a shared filesystem

- We want nodes with the same CPU arch and different interconnect to share the same module tree
- Dynamically swap modules in the OpenMPI stack with **Lmod**

1) Differentiate IB and non-IB modules with a `versionsuffix`

WITH HOOK

You already know how to do it ;)

```
UCX/1.10.0-GCCcore-10.3.0-ib.lua
UCX/1.10.0-GCCcore-10.3.0.lua
UCX/.modulerc.lua
```

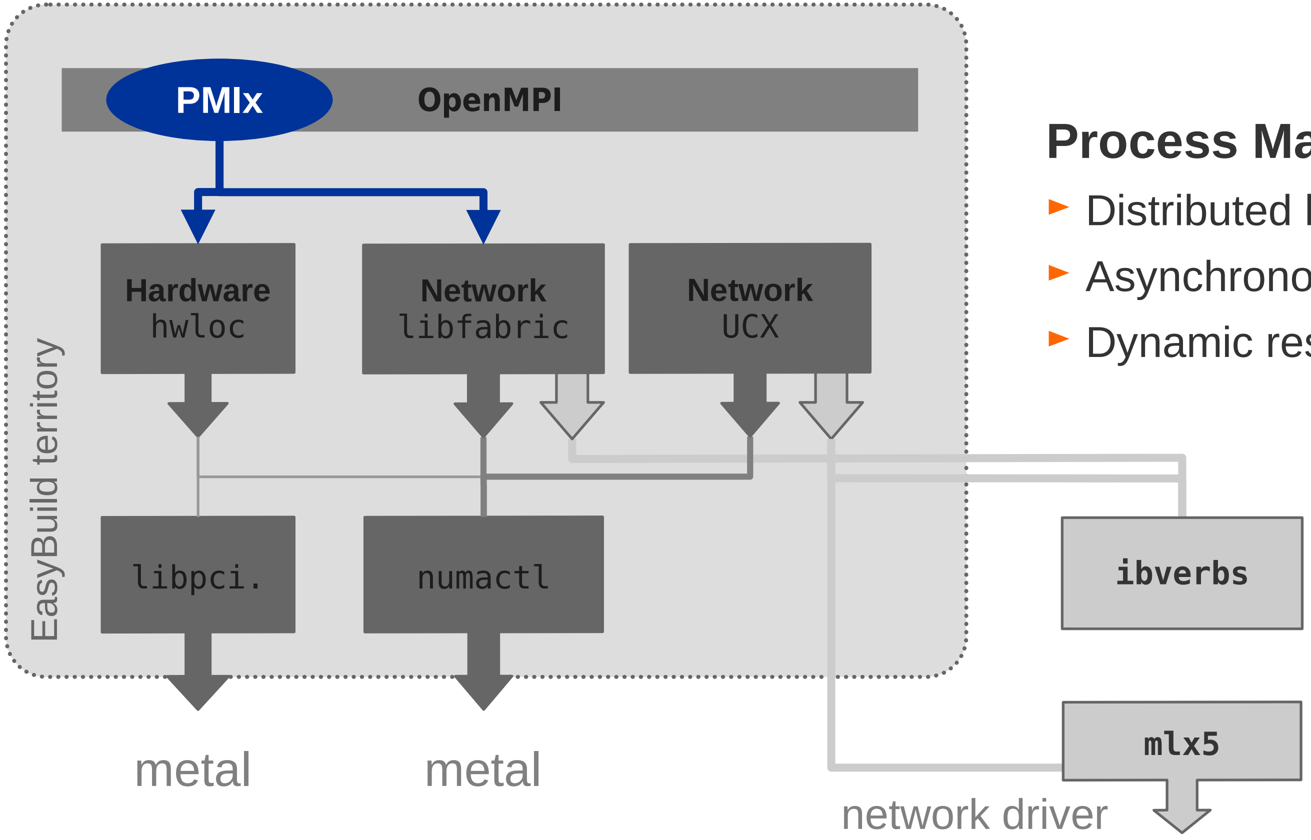
2) Swap the modules on load depending on the system

`.modulerc.lua`

```
if ( os.getenv("NODE_TYPE") == "IB") then
    module_version("UCX/1.10.0-GCCcore-10.3.0-ib", "1.10.0-GCCcore-10.3.0")
end
hide_version("UCX/1.10.0-GCCcore-10.3.0-ib")
```

Lmod
lmod.readthedocs.io

PMIX

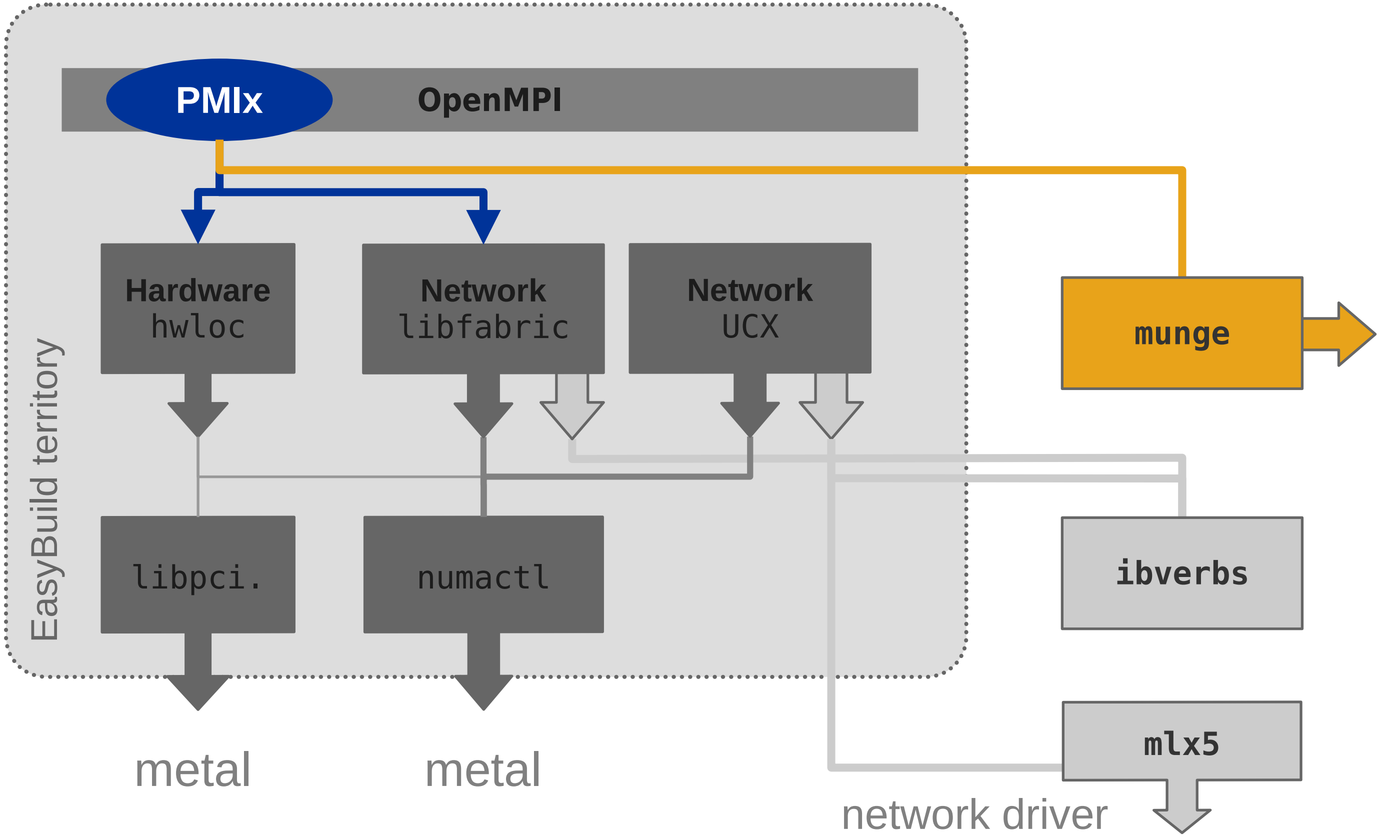


Process Management Interface (PMI)

- Distributed key/value store
- Asynchronous communication
- Dynamic resource management

OpenPMIx
openpmix.github.io

RESOURCE MANAGER



Slurm

- Supports PMIx
- Authentication with **munge**

PMIx

- Add dependency on munge

Slurm
slurm.schedmd.com

PMIX WITH MUNGE

- Enable/disable OS dependency on `munge-devel` in PMIx with EB hooks

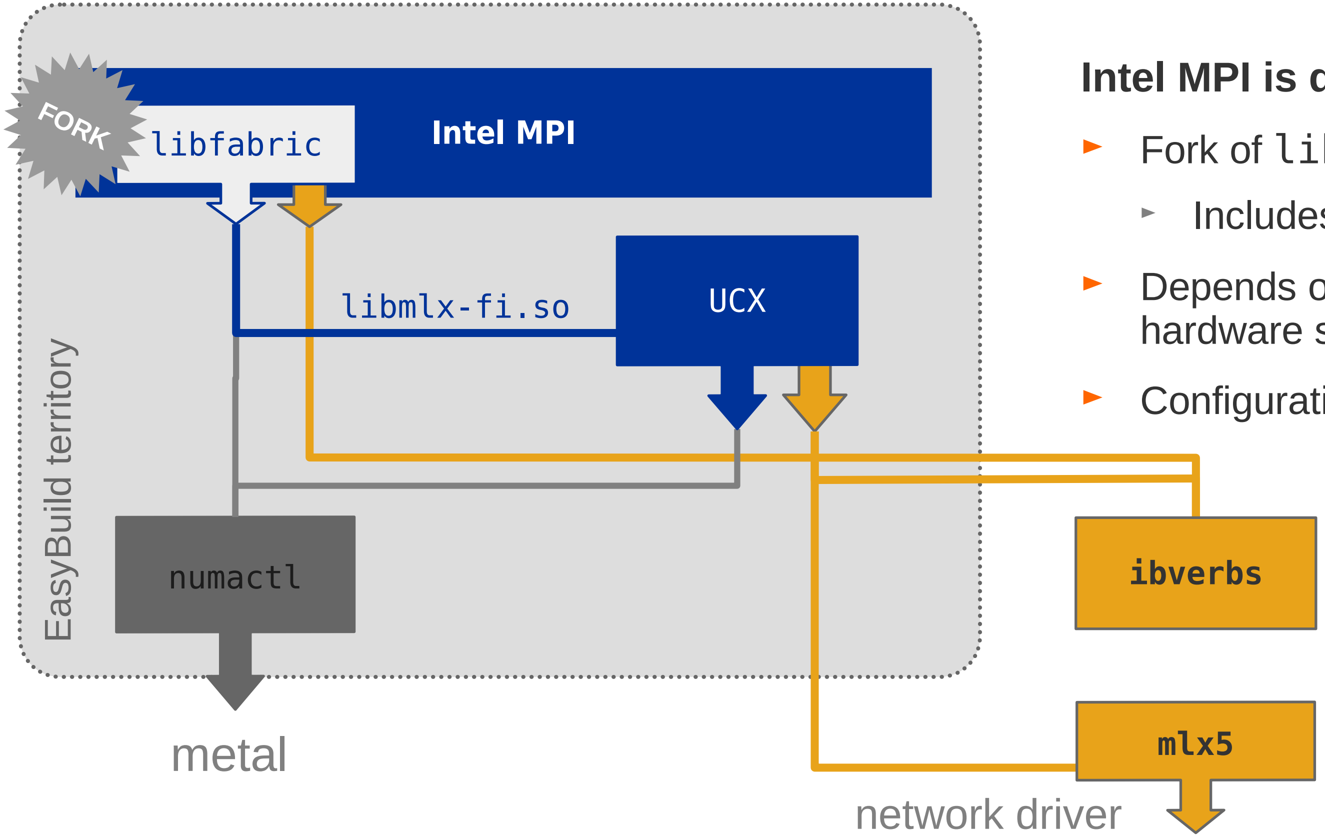
```python
def parse_hook(ec, *args, **kwargs):
    """Alter the parameters of easyconfigs"""

    if ec.name == 'PMIx':
        # Add OS dependency on munge-devel
        extradep = 'munge-devel'
        ec.log.info("[parse hook] Adding OS dependency on: %s" % extradep)
        ec['osdependencies'].append(extradep)
```

More details on our integration of Slurm and PMIx will be shown at **FOSDEM'22** (Feb 6th):

- fosdem.org/2022/schedule/event/exascale_pmi/

INTEL MPI



Intel MPI is distributed as a binary

- Fork of `libfabric` embedded
  - Includes deprecated `mlx` provider
- Depends on external UCX for certain hardware support
- Configuration is mostly done at **runtime**

Intel MPI
intel.com

INTEL MPI CONFIGURATION

- Set runtime settings for Intel MPI with EB hooks

```
from distutils.version import LooseVersion

def pre_module_hook(self, *args, **kwargs):
"Hook at pre-module level to alter module files"

 if self.name == 'impi':
  slurm_mpi_type = None
  intel_mpi = {
   'pmi_var': 'I_MPI_PMI2',
   'pmi_set': 'no',
   'pmi_lib': '/usr/lib64/slurmpmi/libpmi.so',
  }
  if LooseVersion(self.version) >= '2019.7':
   intel_mpi['pmi_var'] = 'I_MPI_PMI'
   intel_mpi['pmi_set'] = 'pmi2'
   intel_mpi['pmi_lib'] =
    '/usr/lib64/slurmpmi/libpmi2.so'
   slurm_mpi_type = 'pmi2'
  # […] add stuff for more versions below
```

DEFAULT SETTINGS

```
  # […] stuff for more versions above

  if slurm_mpi_type:
    self.cfg['modextravars'].update(
      {'SLURM_MPI_TYPE': slurm_mpi_type}
    )

  self.cfg['modluafooter'] = """
if ( os.getenv("SLURM_JOB_ID") ) then
 setenv("I_MPI_HYDRA_BOOTSTRAP", "slurm")
 setenv("I_MPI_PIN_RESPECT_CPUSET", "0")
 setenv("I_MPI_PMI_LIBRARY", "%(pmi_lib)s")
 setenv("%(pmi_var)s", "%(pmi_set)s")
end
""" % intel_mpi
```

CUDA

CUDA also has a role in the MPI stack

- **GPUDirect RDMA** enables direct GPU-GPU communication
- **GDRCopy** improves host to GPU transfers

MPI-CUDA stack in EasyBuild

- **Nvidia drivers**: external to EB
  - minimum version needed for CUDA and GPUDirect RDMA

HOST SYSTEM

- **CUDA**: fully covered in EB
- **UCX-CUDA**: fully covered in EB
  - same approach as with UCX

EASY BUILD

- **GDRCopy**: partially covered in EB
  - EasyBuild installs the libraries, independent of CUDA or the Nvidia drivers
  - But it needs a kernel module to work.

HOST + EB

CONCLUSIONS

- Deploying the MPI stack with EasyBuild needs special care
- Site specific configurations and system libraries have to be checked
- We can automatically apply customizations in EasyBuild thanks to its hooks

As a result, we can install the MPI stack in multiple toolchain generations in a complex hardware environment with our eyes closed and go for a drink! 💙💙

(...and pray that the post-install tests in ReFrame are positive)

ACKNOWLEDGEMENTS

- **Ward Poelmans and Sam Moors (colleagues in VUB-HPC)** ♥ ♥
- EasyBuild community for its openness
- VUB for hosting us and feeding new users to our cluster
- VSC for financial support