Code Migration and Software Environments

Mahidhar Tatineni
HPC and Data Science Summer Institute
San Diego Supercomputer Center

August 4, 2025

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

https://github.com/sdsc/sdsc-summer-institute-2025/tree/main/2.3_code_migration_and_software_environment

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

Applications already available on Expanse

- SDSC staff have installed and made available a large suite of libraries and applications on Expanse. This includes commercially licensed software such as Q-Chem, Abaqus, and Gaussian.
- The primary approach is to make applications available using environment modules.
- Use “module spider” to find applications that are available via modules.
- Some applications/frameworks (e.g. PyTorch, TensorFlow, AlphaFold) have been made available using singularity containers.
- Today’s talk will be focused on Expanse specific options. The options are broadly implementable on the Triton Shared Computing Cluster (TSCC) as the software stack on TSCC is installed w/ a similar approach as Expanse.
- On our Kubernetes based clusters (Voyager, Nautilus/PNRP) we are mainly using a containerized approach (docker images can be used). On Nautilus/PNRP we have a gitlab + Kubernetes development environment and a Coder (https://coder.com/) based development environment.

Applications via modules

- Use “module spider” to find applications.

```
[mahidhar@login01 ~]$ module spider matlab

----------------------------------------------------------------------------------------------------------------
  matlab:
----------------------------------------------------------------------------------------------------------------
     Versions:
        matlab/2020b
        matlab/2022a
     Other possible modules matches:
        matlab/2022b

----------------------------------------------------------------------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*matlab.*'

----------------------------------------------------------------------------------------------------------------
  For detailed information about a specific "matlab" package (including how to load the modules) use the module's full n
ame. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

     $ module spider matlab/2022a
----------------------------------------------------------------------------------------------------------------
```

Applications via modules

- Use “module spider” to find applications.

```
[mahidhar@login01 ~]$ module spider matlab/2022b

----------------------------------------------------------------------------------------------------------------
  matlab/2022b:
----------------------------------------------------------------------------------------------------------------
     Versions:
        matlab/2022b/lefe4oq
        matlab/2022b/nmbr5dd

----------------------------------------------------------------------------------------------------------------
  For detailed information about a specific "matlab/2022b" package (including how to load the modules) use the module's
full name. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

     $ module spider matlab/2022b/nmbr5dd
----------------------------------------------------------------------------------------------------------------
```

Applications via modules

- Use “module spider” to find applications.

```
[mahidhar@login01 ~]$ module spider matlab/2022b/lefe4oq

----------------------------------------------------------------------------------------------------------
  matlab/2022b: matlab/2022b/lefe4oq
----------------------------------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "matlab/2022b/lefe4oq" module is availa
ble to load.

      cpu/0.17.3b

    Help:
      MATLAB (MATrix LABoratory) is a multi-paradigm numerical computing
      environment and fourth-generation programming language. A proprietary
      programming language developed by MathWorks, MATLAB allows matrix
      manipulations, plotting of functions and data, implementation of
      algorithms, creation of user interfaces, and interfacing with programs
      written in other languages, including C, C++, C#, Java, Fortran and
      Python. Note: MATLAB is licensed software. You will need to create an
      account on the MathWorks homepage and download MATLAB yourself. Spack
      will search your current directory for the download file. Alternatively,
      add this file to a mirror so that Spack can find it. For instructions on
      how to set up a mirror, see
      http://spack.readthedocs.io/en/latest/mirrors.html
```

Applications and Libraries via modules

```
[mahidhar@login01 ~]$ module spider fftw

-----------------------------------------------------------------------------------------------------------------
  fftw:
-----------------------------------------------------------------------------------------------------------------
     Versions:
        fftw/2.1.5
        fftw/3.3.8
     Other possible modules matches:
        amdfftw  amdfftw/3.1  fftw/2.1.5  fftw/3.3.10

-----------------------------------------------------------------------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*fftw.*'

-----------------------------------------------------------------------------------------------------------------
  For detailed information about a specific "fftw" package (including how to load the modules) use the module's full nam
e. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

     $ module spider fftw/3.3.8
-----------------------------------------------------------------------------------------------------------------
```

Applications and Libraries via modules

```
[mahidhar@login01 ~]$ module spider fftw/3.3.10/cesmlwb

-----------------------------------------------------------------------------------------------------------------
  fftw/3.3.10: fftw/3.3.10/cesmlwb
-----------------------------------------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "fftw/3.3.10/cesmlwb" module is availab
le to load.

      cpu/0.17.3b  gcc/10.2.0/npcyll4  mvapich2/2.3.7/iyjtn3x

    Help:
      FFTW is a C subroutine library for computing the discrete Fourier
      transform (DFT) in one or more dimensions, of arbitrary input size, and
      of both real and complex data (as well as of even/odd data, i.e. the
      discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which
      is free software, should become the FFT library of choice for most
      applications.
```

Applications and Libraries via modules

```
[mahidhar@login01 ~]$ module reset
Resetting modules to system default. Reseting $MODULEPATH back to system default. All extra directories will be removed
from $MODULEPATH.
[mahidhar@login01 ~]$ module load cpu/0.17.3b
[mahidhar@login01 ~]$ module load gcc/10.2.0/npcyll4  mvapich2/2.3.7/iyjtn3x
[mahidhar@login01 ~]$ module load fftw/3.3.10/cesmlwb
[mahidhar@login01 ~]$ module show fftw/3.3.10/cesmlwb
-------------------------------------------------------------------------------------------------------------
   /cm/shared/apps/spack/0.17.3/cpu/b/share/spack/lmod/linux-rocky8-x86_64/mvapich2/2.3.7-iyjtn3x/gcc/10.2.0/fftw/3.3.10
/cesmlwb.lua:
-------------------------------------------------------------------------------------------------------------
whatis("Name : fftw")
whatis("Version : 3.3.10")
whatis("Target : zen2")
whatis("Short description : FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or
more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discr
ete cosine/sine transforms or DCT/DST). We believe that FFTW, which is free software, should become the FFT library of c
hoice for most applications.")
help([[FFTW is a C subroutine library for computing the discrete Fourier
transform (DFT) in one or more dimensions, of arbitrary input size, and
of both real and complex data (as well as of even/odd data, i.e. the
discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which
is free software, should become the FFT library of choice for most
applications.]])
```

Applications and Libraries via modules

```
[mahidhar@login01 ~]$ module show fftw/3.3.10/cesmlwb
-------------------------------------------------------------------------------------------------------
   /cm/shared/apps/spack/0.17.3/cpu/b/share/spack/lmod/linux-rocky8-x86_64/mvapich2/2.3.7-iyjtn3x/gcc/10.2.0/fftw/3.3.10
/cesmlwb.lua:
-------------------------------------------------------------------------------------------------------
whatis("Name : fftw")
whatis("Version : 3.3.10")
whatis("Target : zen2")
whatis("Short description : FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or
more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discr
ete cosine/sine transforms or DCT/DST). We believe that FFTW, which is free software, should become the FFT library of c
hoice for most applications.")
help([[FFTW is a C subroutine library for computing the discrete Fourier
transform (DFT) in one or more dimensions, of arbitrary input size, and
of both real and complex data (as well as of even/odd data, i.e. the
discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which
is free software, should become the FFT library of choice for most
applications.]])
prepend_path("LD_LIBRARY_PATH","/cm/shared/apps/spack/0.17.3/cpu/b/opt/spack/linux-rocky8-zen2/gcc-10.2.0/fftw-3.3.10-ce
smlwbzqh4xtaiecfmxcbjkjafmpghj/lib")
prepend_path("PATH","/cm/shared/apps/spack/0.17.3/cpu/b/opt/spack/linux-rocky8-zen2/gcc-10.2.0/fftw-3.3.10-cesmlwbzqh4xt
aiecfmxcbjkjafmpghj/bin")
prepend_path("MANPATH","/cm/shared/apps/spack/0.17.3/cpu/b/opt/spack/linux-rocky8-zen2/gcc-10.2.0/fftw-3.3.10-cesmlwbzqh
4xtaiecfmxcbjkjafmpghj/share/man")
```

Hands On! - Command set #1:

Working with modules – Follow instructions in Code-Migration-Handson.md file in the github directory for this session

https://github.com/sdsc/sdsc-summer-institute-2025/tree/main/2.3_code_migration_and_software_environment

Applications via modules

- “module spider” also gives the loading information

- The CPU and GPU stacks are completely independent. Do *not* mix the two as something compiled for the GPU stack will not work on the CPU nodes (which have a different architecture) and vice versa

- Usage examples are provided in:

  /cm/shared/examples/sdsc

```
[mahidhar@login02 sdsc]$ cd /cm/shared/examples/sdsc/
[mahidhar@login02 sdsc]$ ls
abaqus     bintest  dftbplus  gromacs  localscratch       namd    openacc   pyscf    raxml   tensorflow  vasp-ase
abinit     ciml     excerpt   hadoop   matlab             neuron  openmp    pytorch  si      test        visit
alphafold  classes  gamess    hpl      mpi                nsight  orca      qchem    siesta  trinity     wannier90
amber      cp2k     gaussian  lammps   mpi-openmp-hybrid  nwchem  paraview  qe       spark   vasp        xpmem
```

Applications available via Singularity Containers

- Some applications are easier to make available via Singularity containers.
- On Expanse the containers are at:
  - /cm/shared/apps/containers/singularity

- Applications available via Singularity include:
  - TensorFlow, PyTorch, AlphaFold, Paraview, VisIt
  - We also have the Extreme-scale Scientific Software Stack (E4S) available via a container. Ref: https://e4s-project.github.io

- The Singularity definition files are available* for users who wish to add to the containers/rebuild them.

* https://github.com/mkandes/naked-singularity/tree/master/definition-files

Using Applications via Singularity Containers

```
#!/usr/bin/env bash
#SBATCH --job-name=pytorch-gpu-shared
#### Change account below
#SBATCH --account=XYZ123
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=1
#SBATCH --mem=90G
#SBATCH --gpus=1
#SBATCH --time=00:30:00
#SBATCH --output=pytorch-gpu-shared.o%j.%N
module reset
module load singularitypro
time -p singularity exec --bind /expanse,/scratch --nv /cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif python3 $SLURM_SUBMIT_DIR/main.py
```

Building your own Singularity Containers from Existing Docker images

- Most cases you might find an optimized Docker container already exists for the work you want do
- For example, you might have NVIDIA GPU Cloud (NGC) containers OR your organization/community has built custom docker images
- Singularity images can be built from these docker images. Keep in mind
  - The build process might take a lot of temporary space and memory => do this on a compute or GPU node
  - Make sure the image you choose is compatible with drivers on the system. For example, latest NVIDIA image might need a CUDA version that is not supported on Expanse. Note – NGC container release notes also provide compatibility matrix
  - Get interactive access to a node, set TMPDIR to /scratch/$USER/job_$SLURM_JOBID
  - Example build command:

```
singularity build pytorch-nvcr-25.03.sif docker://nvcr.io/nvidia/pytorch:25.03-py3
```

Example of release notes showing driver requirements

https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-25-03.html#rel-25-03

Driver Requirements

Release 25.03 is based on **CUDA 12.8.1.012** which requires **NVIDIA Driver** release 570 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545, R555, and R560 drivers, which are not forward-compatible with CUDA 12.8. For a complete list of supported drivers, see the **CUDA Application Compatibility** topic. For more information, see **CUDA Compatibility and Upgrades**.

docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-25-03.html#rel-25-03

NVIDIA DOCS HUB Topics NVIDIA Developer Blog Forums

Topics

NVIDIA Docs Hub › NVIDIA Optimized Frameworks › NVIDIA Optimized Frameworks › PyTorch Release 25.03

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is prebuilt and installed in the default Python environment (`/usr/local/lib/python3.12/dist-packages/torch`) in the container image.

The container also includes the following:

- **Ubuntu 24.04** including **Python 3.12**
- **NVIDIA CUDA 12.8.1.012**
- **NVIDIA cuBLAS 12.8.4.1**
- **NVIDIA cuBLASMp** 0.4.0
- **NVIDIA cuDNN 9.8.0.87**
- **NVIDIA NCCL 2.25.1**
- NVIDIA RAPIDS™ 25.02
- **rdma-core 50.0**
- NVIDIA HPC-X 2.21
- SHARP 3.9.0
- UCC 1.4.0
- UCX 1.18.0
- **OpenMPI 4.1.7**
- GDRCopy 2.4.1
- TensorBoard 2.16.2
- **Nsight Compute 2025.1.1.2**
- **Nsight Systems 2025.1.1.110**
- **NVIDIA TensorRT™ 10.9.0.34**
- **Torch-TensorRT 2.7.0a0**
- **NVIDIA DALI® 1.47**
- **nvImageCodec 0.2.0.7**
- **MAGMA 2.6.2**
- **JupyterLab 4.3.5** including **Jupyter-TensorBoard**
- **TransformerEngine** 2.1
- TensorRT Model Optimizer 0.23

Driver Requirements

Release 25.03 is based on **CUDA 12.8.1.012** which requires **NVIDIA Driver** release 570 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545, R555, and R560 drivers, which are not forward-compatible with CUDA 12.8. For a complete list of supported drivers, see the **CUDA Application Compatibility** topic. For more information, see **CUDA Compatibility and Upgrades**.

Hands On! - Command set #2

Running using Singularity images

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

Python applications/libraries via modules

- Several python applications and libraries are available via modules
- Examples
  - py-matplotlib, py-numpy, py-scipy
  - py-pysam, py-phonopy, py-htseq
- Use “module spider” and “module show” to get more information
- Some of these modules are dependent on MPI like py-pyscf and py-mpi4py
- Use modules-based approach if all the packages needed are in modules.
- If further installs are needed, its best to go to the conda (Miniforge) or Singularity approach.
- Don’t mix the python apps/libraries from the modules with ones from conda (Miniforge) or a container. They will likely conflict/fail.

Installs using Miniforge or Miniconda3 conda installers

- Use Miniforge or Miniconda3 conda installs if the software available on the system doesn't cover your needs. Specially if custom installs with both python and non-python dependencies are required.
- Install in your home directory. *Do not do conda installs into any Lustre location - can cause systemwide problems due to metadata loads.*
- For GPU installs, make sure the packages chosen work with driver on Expanse. For example, the current versions are:
  - NVIDIA-SMI 525.85.12 Driver Version: 525.85.12 CUDA Version: 12.0
  - Make sure any conda based installs are with a CUDA version that is 12.0 or older.
- https://github.com/conda-forge/miniforge
- Note that Miniforge installs set the conda-forge channel is set as the default (and only) channel. If you are using miniconda3 installs with other channels, please check the updated licensing terms.

Conda Installs using Miniforge

```
[mahidhar_test@login01 ~]$ srun --pty --nodes=1 --ntasks-per-node=1 --cpus-per-t
ask=4 -p shared -A ddp386 -t 01:00:00 --wait 0 /bin/bash
srun: job 34288325 queued and waiting for resources
srun: job 34288325 has been allocated resources
[mahidhar_test@exp-1-01 ~]$ wget https://github.com/conda-forge/miniforge/releas
es/latest/download/Miniforge3-Linux-x86_64.sh
--2024-10-03 09:42:59--  https://github.com/conda-forge/miniforge/releases/lates
t/download/Miniforge3-Linux-x86_64.sh
Resolving github.com (github.com)... 140.82
Connecting to github.com (github.com)|140.8
HTTP request sent, awaiting response... 302
Location: https://github.com/conda-forge/mi
niforge3-Linux-x86_64.sh [following]
--2024-10-03 09:42:59--  https://github.com
oad/24.7.1-2/Miniforge3-Linux-x86_64.sh
```

```
Miniforge3-Linux-x8 100%[===================>]  86.05M   563MB/s    in 0.2s

2024-10-03 09:43:00 (563 MB/s) - 'Miniforge3-Linux-x86_64.sh' saved [90228705/90
228705]

[mahidhar_test@exp-1-01 ~]$ CLEAR
bash: CLEAR: command not found
[mahidhar_test@exp-1-01 ~]$
[mahidhar_test@exp-1-01 ~]$ clear

[mahidhar_test@exp-1-01 ~]$ sh Miniforge3-Linux-x86_64.sh

Welcome to Miniforge3 24.7.1-2

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

- Do the installs on a compute or GPU node.
- wget the installer script

Conda Installs using Miniforge

```
    - cudatoolkit==11.8.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-24.9.1               |  py312h7900ff3_0         1.1 MB  conda-forge
    cudatoolkit-11.8.0         |      h4ba93d1_13       682.5 MB  conda-forge
    ------------------------------------------------------------
                                           Total:       683.6 MB

The following NEW packages will be INSTALLED:

  cudatoolkit        conda-forge/linux-64::cudatoolkit-11.8.0-h4ba93d1_13

The following packages will be UPDATED:

  conda                              24.7.1-py312h7900ff3_0 --> 24.9.1-py312h790
0ff3_0


Proceed ([y]/n)?
```

```
Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: \ By downloading and using the CUDA Toolkit conda package
s, you accept the terms and conditions of the CUDA End User License Agreement (E
ULA): https://docs.nvidia.com/cuda/eula/index.html

done

(base) [mahidhar_test@exp-1-01 ~]$
(base) [mahidhar_test@exp-1-01 ~]$ conda list
# packages in environment at /home/mahidhar_test/miniforge3:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                 conda_forge    conda-forge
_openmp_mutex             4.5                       2_gnu    conda-forge
archspec                  0.2.3             pyhd8ed1ab_0    conda-forge
boltons                   24.0.0            pyhd8ed1ab_0    conda-forge
```

mpi4py

- mpi4py install needs to be consistent with the MPI being used. The system installed versions:

```
[mahidhar@login01 ~]$ module spider py-mpi4py/3.1.2/cllp7nt

-----------------------------------------------------------------------------------------------------------
  py-mpi4py/3.1.2: py-mpi4py/3.1.2/cllp7nt
-----------------------------------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "py-mpi4py/3.1.2/cllp7nt" module is ava
ilable to load.

      cpu/0.17.3b  gcc/10.2.0/npcyll4  intel-mpi/2019.10.317/kdx4qap

    Help:
      This package provides Python bindings for the Message Passing Interface
      (MPI) standard. It is implemented on top of the MPI-1/MPI-2
      specification and exposes an API which grounds on the standard MPI-2 C++
      bindings.
```

```
[mahidhar@login01 ~]$ module spider py-mpi4py/3.1.2/silsqln

-----------------------------------------------------------------------------------------------------------
  py-mpi4py/3.1.2: py-mpi4py/3.1.2/silsqln
-----------------------------------------------------------------------------------------------------------

    You will need to load all module(s) on any one of the lines below before the "py-mpi4py/3.1.2/silsqln" module is ava
ilable to load.

      cpu/0.17.3b  gcc/10.2.0/npcyll4  mvapich2/2.3.7/iyjtn3x

    Help:
      This package provides Python bindings for the Message Passing Interface
      (MPI) standard. It is implemented on top of the MPI-1/MPI-2
      specification and exposes an API which grounds on the standard MPI-2 C++
      bindings.
```

- Note that if you do a conda install of mpi4py, it will use a conda based mpi install (and not the system one). This is ok for single node cases, but multi-node will not use the high-performance InfiniBand network.
- Use system installed MPI and combine with the conda install by building mpi4py from source.

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

R applications/libraries via modules

- Several R applications and libraries are available via modules
- Use “module spider r-” to find all installed apps/libraries.
- Examples
  - r-biobase, r-deseq2, r-doparallel, r-ggplot2
- Use modules-based approach if all the packages needed are in modules.
- If further installs are needed, its best to go to the Miniforge conda install or Singularity approach.
- Don’t mix the R apps/libraries from the modules with ones from conda installs or a container. They will likely conflict/fail.

Example of using R via Singularity

```
[xdtr112@exp-9-55 ~]$ export TMPDIR=/scratch/$USER/job_$SLURM_JOBID
[xdtr112@exp-9-55 ~]$ module load singularitypro
[xdtr112@exp-9-55 ~]$ singularity build excerpt.sif docker://rkitchen/excerpt
WARNING: 'nodev' mount option set on /scratch, it could be a source of failure during build process
INFO:    Starting build...
Getting image source signatures
Copying blob 5e35d10a3eba done
Copying blob cc17f052e960 done
Copying blob 6059ce0d04ff done
Copying blob be036d7c9474 done
Copying blob a59ce8f0c359 done
Copying blob 236bb8549592 done
Copying blob e6adcfd80d7c done
```

R via conda install

```
(base) [mahidhar_test@login02 ~]$ conda search r-base==4*
Loading channels: done
# Name                       Version           Build  Channel
r-base                         4.0.0      hdca8982_2  conda-forge
r-base                         4.0.0      hdca8982_3  conda-forge
r-base                         4.0.1      h95c6c4b_0  conda-forge
r-base                         4.0.2      h95c6c4b_0  conda-forge
r-base                         4.0.2      he766273_1  conda-forge
r-base                         4.0.3      h349a78a_8  conda-forge
r-base                         4.0.3      h8ff2632_7  conda-forge
r-base                         4.0.3      ha43b4e8_3  conda-forge
r-base                         4.0.3      hd23ff56_4  conda-forge
r-base                         4.0.3      hd23ff56_5  conda-forge
r-base                         4.0.3      hd23ff56_6  conda-forge
r-base                         4.0.5      h06d3f91_6  conda-forge
r-base                         4.0.5      h8cab1ac_0  conda-forge
r-base                         4.0.5      h9e01966_1  conda-forge
r-base                         4.0.5      ha8c3e7c_7  conda-forge
r-base                         4.0.5      hb67fd72_2  conda-forge
r-base                         4.0.5      hb87df5d_8  conda-forge
r-base                         4.0.5      hb93adac_3  conda-forge
r-base                         4.0.5      hd930d0e_5  conda-forge
```

- **Do *not* do installs of R packages from the login nodes. R installs tend to try and grab all the cores on a node and that will cause problems on the login nodes**
- **You can do package searches using conda commands. This is fine on a login node**

Example of using R via Singularity

```
#!/bin/bash
#SBATCH --job-name="excerpt-test"
#SBATCH --output="excerpt.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=16G
#SBATCH --account=XYZ123
#SBATCH -t 04:00:00

### Modules
module reset
module load singularitypro

### Run the job
singularity run --bind $SLURM_SUBMIT_DIR/input:/exceRptInput --bind $SLURM_SUBMIT_DIR/output:/exceRptOutput --bind
/expanse/projects/qstore/data/excerpt/hg38:/exceRpt_DB/hg38 /cm/shared/apps/containers/singularity/excerpt/excerpt.sif
INPUT_FILE_PATH=/exceRptInput/SRR026761.sra
```

Hands On! - Command set #3

Build a Singularity image for a R application using a docker image

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

Installs from source, configure example

- Some applications have configure scripts that pick up dependencies based on environment variables and options.
- Example: Quantum Espresso
- Build using a job script:
  - /cm/shared/examples/sdsc/qe/cpu_stack_0.15.4/build_scripts/qe_gcc92_openmpi.sh
- Compile environment:

```
module reset
module load cpu/0.15.4
module load gcc/9.2.0
module load openmpi/3.1.6
module load amdblis
module load amdlibflame
module load amdfftw
```

Installs from source, QE example continued

- Compiler flags

```
CC=gcc
CXX=g++
F77=gfortran
FC=gfortran
F90=gfortran
export CFLAGS="-O3 -march=core-avx2"
export CXXFLAGS="-O3 -march=core-avx2"
export FCFLAGS="-O3 -march=core-avx2  "
export F90FLAGS="-O3 -march=core-avx2 -cpp "
export F77FLAGS="-O3 -march=core-avx2  "
export FFLAGS="-O3 -march=core-avx2 "
```

Installs from source, QE example continued

- Environment variables

```
export FFT_LIBS="-L${AMDFFTWHOME}/lib -lfftw3"
export FFT_INCLUDE="-I{AMDFFTWHOME}/include"
export FFTW_INCLUDE="-I{AMDFFTWHOME}/include"
export BLAS_LIBS="-L${AMDBLISHOME}lib -lblis"
export LAPACK_LIBS="-L${AMDLIBFLAMEHOME}/lib -lflame"
export IFLAGS="-I../include -I${AMDFFTWHOME}/include -I${AMDBLISHOME}/include -
I${AMDLIBFLAMEHOME}/include"
export SCALAPACK_LIBS="-L${HOME}/scalapack/lib -lscalapack"
```

Installs from source, QE example continued

```
./configure \
   CC=$CC \
   CXX=$CXX \
   F77=$F77 \
   FC=$FC \
   F90=$FC \
   --prefix=$HOME/qe

cp make.sys make.sys.bak
make $ESPRESSO_TARGETS
make install
```

Installs from source, RAxML Makefile example

```
# Makefile November 2009 by Alexandros Stamatakis

CC = mpicc

CFLAGS = -D_WAYNE_MPI -D__SIM_SSE3 -O2 -D_GNU_SOURCE -msse3 -fomit-frame-pointer
 -funroll-loops -D__AVX

LIBRARIES = -lm

RM = rm -f

objs    = axml.o  optimizeModel.o multiple.o searchAlgo.o topologies.o parsePart
itions.o treeIO.o models.o bipartitionList.o rapidBootstrap.o evaluatePartialGen
ericSpecial.o evaluateGenericSpecial.o newviewGenericSpecial.o makenewzGenericSp
ecial.o   classify.o fastDNAparsimony.o fastSearch.o leaveDropping.o rmqs.o rogu
eEPA.o ancestralStates.o avxLikelihood.o  mem_alloc.o eigen.o

all : clean raxmlHPC-MPI-AVX2

GLOBAL_DEPS = axml.h globalVariables.h rmq.h rmqs.h #mem_alloc.h

raxmlHPC-MPI-AVX2 : $(objs)
        $(CC) -o raxmlHPC-MPI-AVX2 $(objs) $(LIBRARIES) $(LDFLAGS)

avxLikelihood.o : avxLikelihood.c $(GLOBAL_DEPS)
                $(CC) $(CFLAGS) -mavx2 -D_FMA -march=core-avx2 -c -o avxLikeliho
od.o avxLikelihood.c

fastDNAparsimony.o : fastDNAparsimony.c $(GLOBAL_DEPS)
                   $(CC) $(CFLAGS) -mavx -c -o fastDNAparsimony.o fastDNAparsimo
```

Installs from source

- All GPU compiles *must* be done on a GPU node. Also, don't mix installs from GPU and CPU stack
- Compiles done on the login node will fail on a GPU node
- CPU codes can be compiled on the login node as the processor matches
- Note on BLAS/LAPACK/SCALAPACK: there are several options
  - OpenBLAS,
  - Netlib Scalapack
  - MKL
  - AOCL
- Libraries will be in non-standard locations. So, make sure to use configure/cmake/makefile options to point build scripts to the right locations
- Intel compilers work fine on AMD nodes. Don't use "-xHOST", switch to "-march=core-avx2"

Hands On! - Command set #4

Downloading and building RAxML code using a Makefile

Installs from source – Spack based installs

- Spack is a package manager that makes is it easy to install scientific software on HPC systems https://spack.io/
- SDSC maintains scripts and spack package repositories for machine specific installs https://github.com/sdsc/spack
- AMD provides performant Spack recipes and specs: https://www.amd.com/en/developer/zen-software-studio/applications/spack.html

```
module reset
git clone --depth=100 --branch=releases/v0.22 https://github.com/spack/spack.git ~/spack
. $HOME/spack/share/spack/setup-env.sh
spack compiler find
spack env create vasp-aocc
spack env activate vasp-aocc
spack install --add aocc@4.2.0 +license-agreed
spack compiler add
spack install --add openmpi@4.1.6 % aocc@4.2.0 ~atomics~cuda~cxx~cxx_exceptions~gpfs~internal-hwloc~java+legacylaunchers+lustre~memchecker+pmi+romio~rsh~singularity+static+vt+wrapper-rpath fabrics=ucx,cma schedulers=slurm ^ucx@1.14.0 ^lustre@2.15.4 ^slurm@23.02.7 ^rdma-core@58
spack install vasp +scalapack +openmp +fftlib %aocc ^amdfftw ^amdblis threads=openmp ^amdlibflame ^amdscalapack ^openmpi
```

Outline

- Applications already available on Expanse (and other systems)
  - modules
  - singularity containers
- Python based applications/libraries
  - SDSC installed and available via modules
  - Miniforge/Miniconda3 installs
  - mpi4py
- R based applications
  - SDSC installed and available via modules
  - Miniforge, Singularity approaches
- Installing/building applications from source code
  - Builds that utilize dependencies already installed on Expanse
  - Spack based approach for custom environments
- User built containers
  - Singularity, Docker options

Building Singularity containers

- Useful if there are a lot of dependencies that cannot be easily installed in the regular Expanse environment
- For MPI based installs:
  - The MPI in the container should match the external MPI version
  - Make sure the InfiniBand drivers are installed in the container
  - We have example definition files available
- For GPU installs:
  - Make sure packages installed are compatible with driver on our system
- Builds from definition files need root access and cannot be done on Expanse. Build elsewhere - for example use your laptop/desktop OR a cloud resource (e.g. Jetstream2 on ACCESS)

Sample Singularity definition file

```
Bootstrap: shub
From: mkandes/naked-singularity:centos-7.9.2009

%labels

    APPLICATION_NAME centos + mvapich
    APPLICATION_VERSION 7.9.2009 + 2.3.2
    APPLICATION_URL https://mvapich.cse.ohio-state.edu

    AUTHOR_NAME Marty Kandes
    AUTHOR_EMAIL mkandes@sdsc.edu

    LAST_UPDATED 20201227

%setup

%environment

    # Set paths to MVAPICH2 binaries and libraries
    export PATH="/opt/mvapich2-2.3.2/bin:${PATH}"
    export LD_LIBRARY_PATH="/opt/mvapich2-2.3.2/lib:${LD_LIBRARY_PATH}"

%post -c /bin/bash

    # Set operating system mirror URL
    export MIRRORURL='http://mirror.centos.org/centos-7/7.9.2009/os/x86_64'

    # Set operating system version
    export OSVERSION='7'

    # Set system locale
    export LC_ALL=C

    # Update all software packages to their latest versions
    yum -y check-update && yum -y update

    # Install basic drivers for user space access to Ethernet, RDMA,
    # and Infiniband. See https://community.mellanox.com/docs/DOC-2431
    yum -y install dkms
    yum -y install infiniband-diags
    yum -y install infiniband-diags-devel
    yum -y install libibverbs
```

```
    yum -y install libibverbs-devel
    yum -y install ibacm
    yum -y install librdmacm
    yum -y install librdmacm-devel
    yum -y install libmlx4
    yum -y install libmlx5
    yum -y install mstflint
    yum -y install libibcm
    yum -y install libibmad
    yum -y install libibmad-devel
    yum -y install libibumad
    yum -y install libibumad-devel
    yum -y install opensm
    yum -y install srptools

    # Install additional tools
    yum -y install ibutils
    yum -y install libibverbs-utils
    yum -y install librdmacm-utils
    yum -y install perftest
    yum -y install numactl

    # Install libnl
    yum -y install libnl3
    yum -y install libnl3-devel

    # Install mvapich2 (build) dependencies
    yum -y install bison

    cd /tmp

    # Download, build, and install mvapich2
    wget http://mvapich.cse.ohio-state.edu/download/mvapich/mv2/mvapich2-2.3.2.tar.gz
    tar -xzvf mvapich2-2.3.2.tar.gz
    cd mvapich2-2.3.2
    ./configure --prefix=/opt/mvapich2-2.3.2
    make
    make install

    # Cleanup
    package-cleanup -q --leaves | xargs -l1 yum -y remove
    yum -y clean all
```

Conclusions

- Several options to get applications working on Expanse
- Check if application is already installed - either via modules or in Singularity images
- Examples directory (/cm/shared/examples/sdsc)
- Several options for R and python: 1) installs available via modules; 2) miniconda3/miniforge installations in user directories; 3) containers
- Do not mix software installations on system with conda based installs - try to keep the entire application tree needed for a workflow in one environment.
- Build from source – using compilers/libraries already installed; using Spack based approach
- Can use docker images via Singularity on Expanse, TSCC clusters
- Build your own containers from definition files
  - keep InfiniBand stack in the container consistent with the one used on system
  - GPU application/library installs must be compatible with drivers on the system
- Kubernetes clusters (Voyager, Nautilus/PNRP) use containerized environments. Multiple development environments available.