Agenda

- 8:30am Welcome
- 8:35am Shaheen III Hardware Overview
- 8:55am How to apply on Shaheen III
- 9:05am Getting Started on Shaheen III
- **9:15am Software Environment**
- 9:35am Job Scheduling
- 10:00am Coffee Break
- 10:15am Storage overview & Best practices
- 10:30am Applications software example: VASP workflow
- 10:50 am Applications software example: CFD applications
- 11:10 am Applications software example: Bio informatics workflow
- 11:20-11.30am Q&A and Open Discussion

Shaheen III Survey

Shaheen III HPC Training Software Environment

Kadir Akbudak, KAUST Supercomputing Lab

February 4, 2025

Location: Auditorium 0215, Between buildings 2 and 3

Software Environment on Shaheen III

Software supported by KSL and HPE/Cray

- Compilers
- Vendor optimized libraries (math, IO)
- Applications

Self-service software

- Follow recipes from experts: Conda, Containers, community code (WRF, etc.)

- Use separate scripts (other than `.bashrc`) to initialize your environment.

Modules

https://docs.hpc.kaust.edu.sa/soft_env/prog_env/modulesystem/shaheen3/index.html

- To find the list of all the packages installed:

```
module avail
```

- To find a specific package:

```
module avail -S name
```

- To get information on the package usage:

```
module help <package-name>
module show <package-name>
```

- To load a module:

```
module load <package-name>
```

Do not `module purge`

Compiler Toolchains

Default PrgEnv when you login

Vendor	Programming environment	Module	Language	Compiler wrapper	Compiler
Cray	PrgEnv-cray	cce	C	cc	craycc
			C++	CC	crayCC
			Fortran	ftn	crayftn
GNU	PrgEnv-gnu	gcc	C	cc	gcc-13
			C++	CC	g++-13
			Fortran	ftn	gfortran-13
AMD	PrgEnv-aocc	aocc	C	cc	amdclang
			C++	CC	amdclang++
			Fortran	ftn	amdflang
Intel	PrgEnv-intel	intel	C	cc	icx
			C++	CC	icpx
			Fortran	ftn	ifort

Getting new versions of Cray Programming Environment (cpe)

```
module av -S cpe
cpe/23.09     cpe/23.12    cpe/24.07(default)      [cpe/24.11  after Feb 12]

cc --version
Cray clang version 18.0.0  (0e4696aa65fa9549bd5e19c216678cc98185b0f7)

module load cpe/23.12
Switching to PrgEnv-cray/8.5.0.
Switching to cce/17.0.0.
Switching to cray-dsmml/0.2.2.
Switching to cray-libsci/23.12.5.
Switching to cray-mpich/8.1.28.
Switching to craype/2.7.30.
Switching to perftools-base/23.12.0.

cc --version
Cray clang version 17.0.0  (b59b7a8e9169719529cf5ab440f3c301e515d047)
```

Compiler Wrappers

- Switch between compiler toolchains:

```
module switch PrgEnv-cray PrgEnv-gnu
```

- Change the compiler for PrgEnv-cray:

```
module av cce
cce/16.0.1          cce/17.0.0          cce/18.0.0(default)

module switch cce cce/17.0.0
```

Cray MPICH is provided by default. It is also linked when the compiler wrappers cc, CC, and ftn are used.

```
cc -craype-verbose
clang -march=znver4 -dynamic

cc --cray-print-opts=all
-I/opt/cray/pe/mpich/8.1.30/ofi/cray/17.0/include -I/opt/cray/pe/dsmml/0.3.0/dsmml//include
-I/opt/cray/pe/libsci/24.07.0/CRAY/17.0/x86_64/include
-I/opt/cray/xpmem/2.8.4-1.0_7.2__ga37cbd9.shasta/include -L/opt/cray/pe/mpich/8.1.30/ofi/cray/17.0/lib
-L/opt/cray/pe/dsmml/0.3.0/dsmml//lib -L/opt/cray/pe/libsci/24.07.0/CRAY/17.0/x86_64/lib
-L/opt/cray/pe/cce/18.0.0/cce/x86_64/lib/pkgconfig/../ -L/opt/cray/xpmem/2.8.4-1.0_7.2__ga37cbd9.shasta/lib64
-Wl,--as-needed,-lsci_cray_mpi,--no-as-needed -Wl,--as-needed,-lsci_cray,--no-as-needed -ldl
-Wl,--as-needed,-lmpi_cray,--no-as-needed -Wl,--as-needed,-ldsmml,--no-as-needed -lxpmem
-Wl,--as-needed,-lstdc++,--no-as-needed -Wl,--as-needed,-lpgas-shmem,--no-as-needed -lquadmath -lmodules -lfi
-lcraymath -lf -lu -lcsup
```

List of Some Vendor Optimized Libraries

MPI:

- cray-mpich
- mpixlate (using non-binary compatible with HPE Cray MPI)

Math

- cray-libsci, cray-fftw, MKL, AOCL
- cray-python, cray-R

IO

- cray-hdf5, cray-netcdf, iobuf

Environment Variables & Self-Service Software

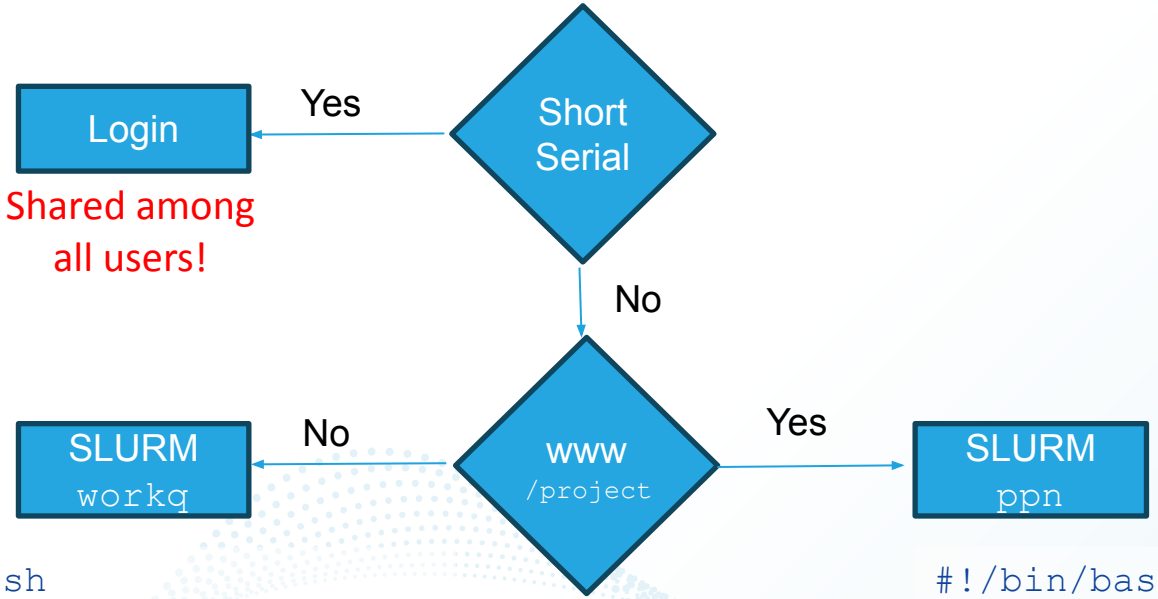
`env` command shows the predefined variables that you can use in your shell scripts.

```
DEFAULT_PROJECT=kXXXXX
PROJECT_DIR=/project/$DEFAULT_PROJECT/$USER

SCRATCH=/scratch/$USER
SCRATCH_BW=/scratch/$USER/bandwidth
SCRATCH_IOPS=/scratch/$USER/iops

MY_SW=/scratch/$USER/iops/sw
MY_SINGULARITY_IMAGES=/scratch/$USER/iops/sw/images

CONDA_PKGS_DIRS=/scratch/$USER/iops/sw/cache
```

Self-Service Software: Compiling A Sample Application

```
cd $PROJECT_DIR
cc hello.c
CC hello.cpp
ftn hello.f


CC=cc CXX=CC FC=ftn cmake ..
make CC=cc CXX=CC
```

Self-Service Software: Compiling A Sample Application




```
#!/bin/bash
#SBATCH --time 12:00:00
#SBATCH --partition=workq

make -j 64
```

```
#!/bin/bash
#SBATCH --time 12:00:00
#SBATCH --partition=ppn

make -j 64
```

Sample build scripts using SLURM:
https://github.com/kaust-rccl/build-recipes/blob/main/WRF/compile.slurm

Self-Service Software: Conda

https://docs.hpc.kaust.edu.sa/soft_env/prog_env/python_package_management/conda/shaheen3.html

```
mkdir -p $MY_SW && cd $MY_SW
bash
/sw/sources/miniconda/conda24.1.2-python3.12.1/Miniconda3-latest-Linux-x86_6
4.sh -b -s -p $MY_SW/miniconda3-amd64 -u
source $MY_SW/miniconda3-amd64/bin/activate
conda install -y -c conda-forge mamba
conda deactivate
```

```
#!/bin/bash
#SBATCH -t 00:10:00
#SBATCH -p workq
source $MY_SW/miniconda3-amd64/bin/activate $MY_SW/envs/pytorch
python -c 'import torch; print("Pytorch Version:",torch.__version__)'
python -c 'import torch; print("Pytorch location:",torch.__file__)'
```

Self-Service Software: Singularity

https://docs.hpc.kaust.edu.sa/soft_env/prog_env/containers/quick_start_singularity_shaheen3.html

A singularity image can be pulled as follows:

```
module load singularity
cd $HOME && mkdir -p tmpdir
export SINGULARITY_TMPDIR=$HOME/tmpdir
singularity pull docker://krccl/cdo_gnu:1.9.10
mkdir -p $MY_SINGULARITY_IMAGES
cp ~/cdo_gnu_1.9.10.sif $MY_SINGULARITY_IMAGES
cd $MY_SINGULARITY_IMAGES
singularity run cdo_gnu_1.9.10.sif cdo --version
```

The following SLURM job script runs the container:

```
#SBATCH --nodes=1
#SBATCH --hint=nomultithread
#SBATCH --time=00:10:00
module load singularity
singularity run cdo_gnu_1.9.10.sif cdo --version
```

Debugging and Profiling

Debugging and profiling tools are available:

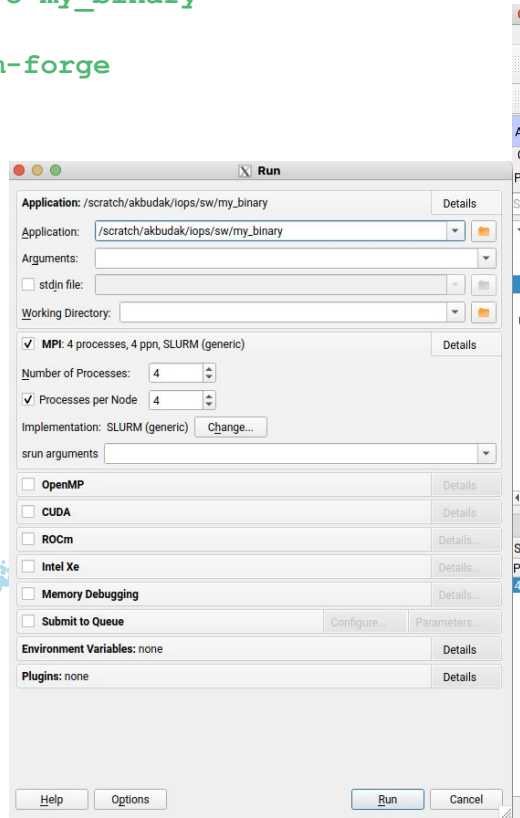
- Linaro Forge, TotalView, gdb4hpc, valgrind4hpc, AMD µProf
- Compile with -g

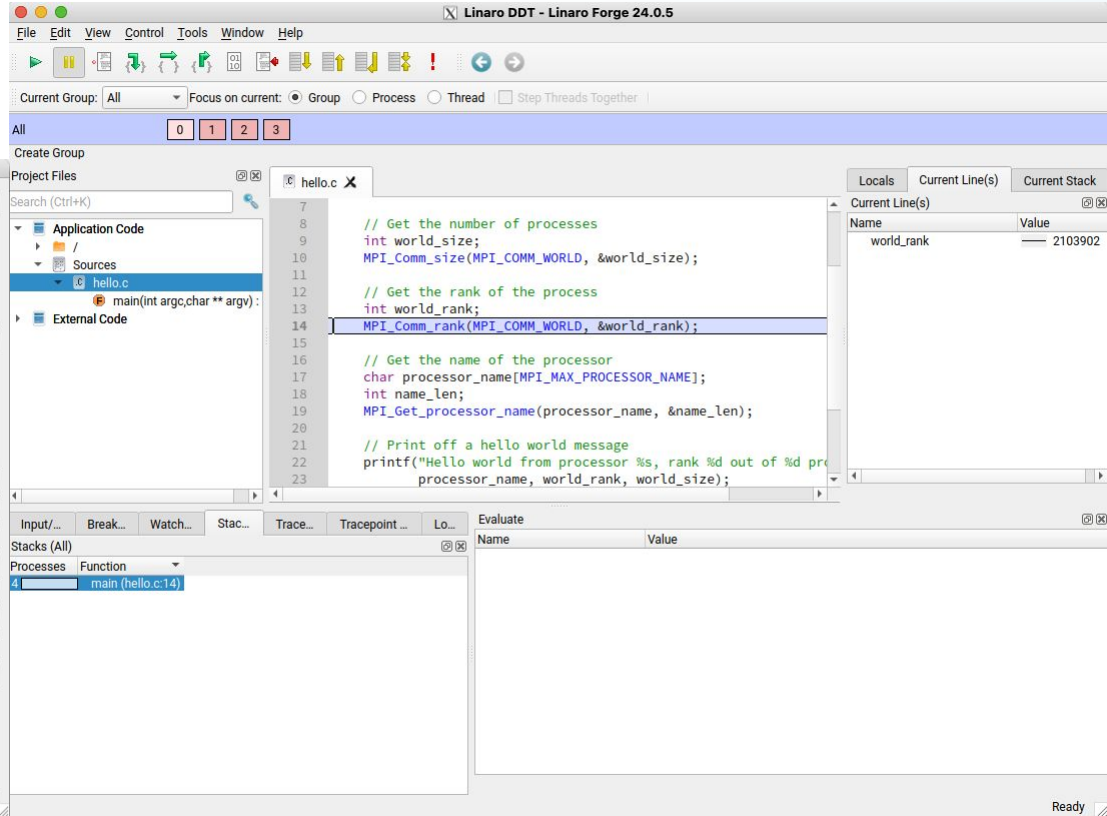
https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/debugging
https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/profiling/

Debugging: Linaro DDT

Step by step debugging many MPI ranks.

```
cc -g hello.c -o my_binary
salloc -N 1
module load arm-forge
ddt &
```

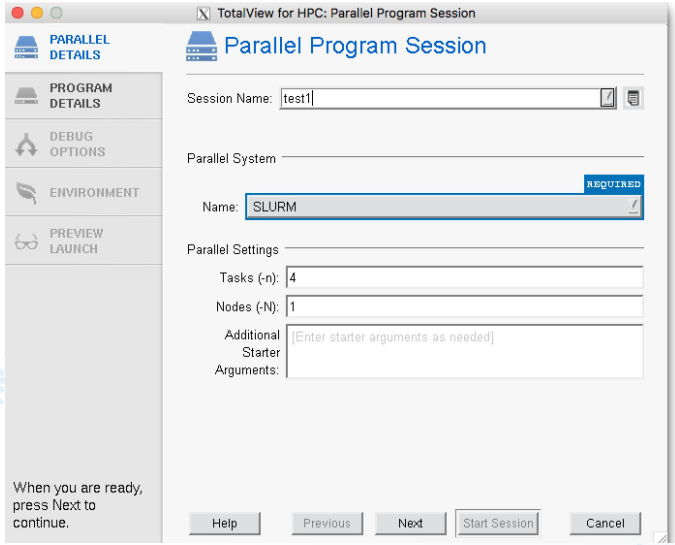


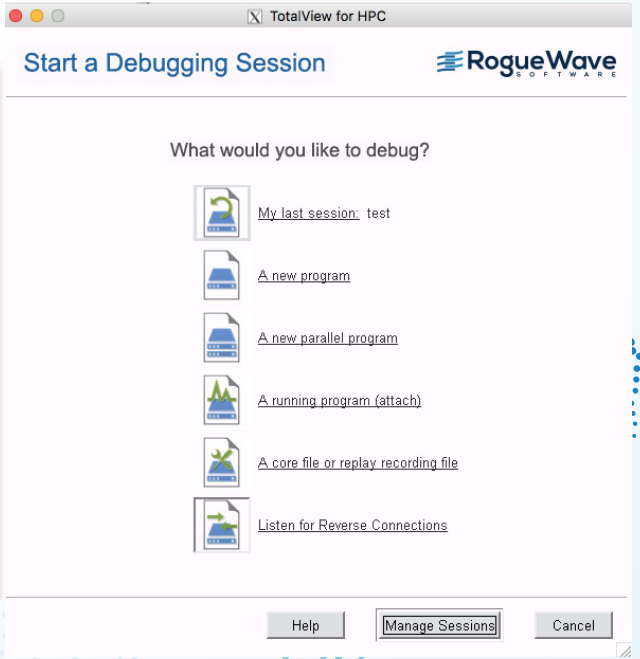


Debugging: TotalView

Allocate the node

```
module load totalview
tv8 &
```





Debugging: gdb4hpc

https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/debugging/gdb4hpc.html

- Step by step debugging many MPI ranks.
- Good for finding problems in your source code.

```
cc -g hello.c -o my_binary
module load gdb4hpc
gdb4hpc

gdb4hpc - Cray Line Mode Parallel Debugger
…
dbg all> launch $a{2} --launcher-args="-N2" ./my_binary
Starting application, please wait...
Launched application...
0/2 ranks connected... (timeout in 300 seconds)
2/2 ranks connected.
Created network...
Connected to application...
Launch complete.
a{0..1}: Initial breakpoint, main at /scratch/akbudak/iops/sw/hello.c:6
dbg all> bt
a{0..1}: #0  main at /scratch/akbudak/iops/sw/hello.c:6
dbg all> c
dbg all> <$a>: Hello world from processor nid00867, rank 0 out of 2 processors
<$a>: Hello world from processor nid00880, rank 1 out of 2 processors
a{0..1}: The application has reached an exit breakpoint.
```

Debugging: valgrind4hpc

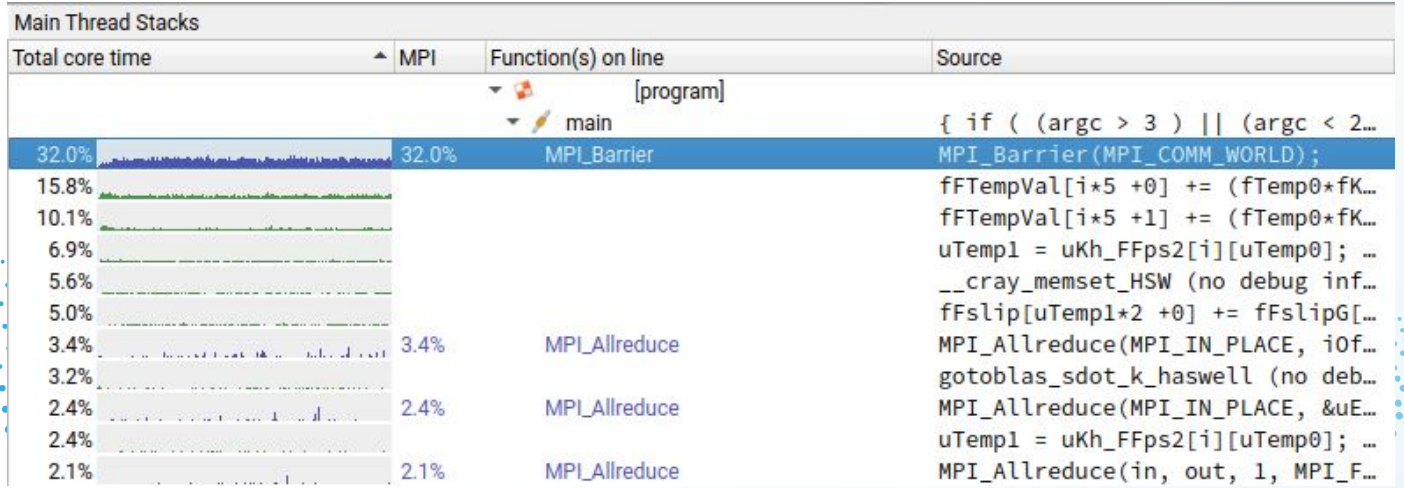
https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/debugging/valgrind4hpc.html

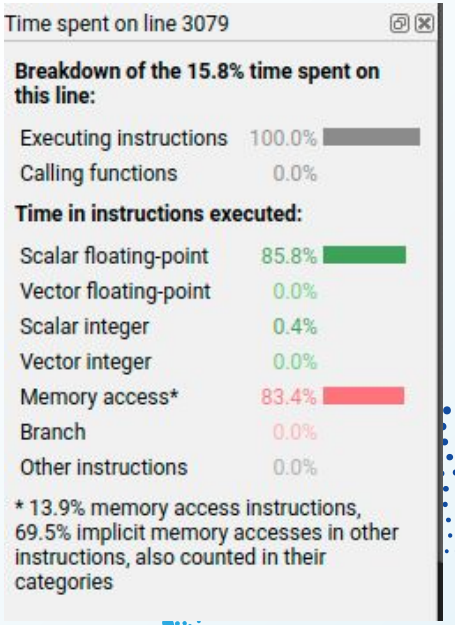
- Memory leaks can be detected using valgrind4hpc.

```
module load valgrind4hpc
valgrind4hpc --valgrind-args="--vgdb=no" -n8 --launcher-args="-N2"
--outputfile=out.txt ./my_binary

RANKS: <0..7>

HEAP SUMMARY:
  in use at exit: 0 bytes in 0 blocks

All heap blocks were freed -- no leaks are possible

ERROR SUMMARY: 179 errors from 722 contexts (suppressed 4443)
```

Profiling: Linaro MAP

```
module load arm-forge
make-profiler-libraries
cc -g hello.c -o my_binary -dynamic -L/scratch/akbudak/iops/sw
-lmap-sampler-pmpi -lmap-sampler -Wl,--eh-frame-hdr
-Wl,-rpath=/scratch/akbudak/iops/sw -ldl

perf-report srun -n 2 ./my_binary
```





Profiling: CrayPAT

https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/profiling/craypat.html

```
module load perftools-lite
cc hello.c
WARNING: PerfTools is saving object files from a temporary directory into directory
'/home/akbudak/.craypat/a.out/1731317'
INFO: creating the PerfTools-instrumented executable 'a.out' (lite-samples) ...OK
srun -n 8 ./a.out
Avg Process Time:   0.05 secs
High Memory:        318.0 MiBytes  39.7 MiBytes per PE
I/O Read Rate:           -- MiBytes/sec
I/O Write Rate:   6.857979 MiBytes/sec

For a complete report with expanded tables and notes, run:
  pat_report /scratch/akbudak/iops/sw/a.out+2538194-1753737628s

For help identifying callers of particular functions:
  pat_report -O callers+src /scratch/akbudak/iops/sw/a.out+2538194-1753737628s
To see the entire call tree:
  pat_report -O calltree+src /scratch/akbudak/iops/sw/a.out+2538194-1753737628s

For interactive, graphical performance analysis, run:
  app2 /scratch/akbudak/iops/sw/a.out+2538194-1753737628s
```

Profiling: AMD µProf

https://docs.hpc.kaust.edu.sa/soft_env/prof_debug/profiling/uprof.html

- Roofline analysis: Does my workflow use the CPU and memory resources efficiently? Is there room for improvement?
- Analyze the performance of one or more processes or the entire system.
- Performance bottlenecks (hotspots & micro-architecture) in the source code.
- Optimize the source code for better performance and power efficiency.

```
salloc -N 1
srun --pty /bin/bash
module load amduprof
cd /scratch/$USER/iops/sw
AMDuProfPcm roofline  -o ./roofline.csv ./a.out

module load python
AMDuProfModelling.py -i ./roofline.csv -o . --memspeed 4800 -a MyHelloWorld --memory-roofs all
```

Agenda

- 8:30am Welcome
- 8:35am Shaheen III Hardware Overview
- 8:55am How to apply on Shaheen III
- 9:05am Getting Started on Shaheen III
- 9:15am Software Environment
- **9:35am Job Scheduling**
- 10:00am Coffee Break
- 10:15am Storage overview & Best practices
- 10:30am Applications software example: VASP workflow
- 10:50 am Applications software example: CFD applications
- 11:10 am Applications software example: Bio informatics workflow
- 11:20-11.30am Q&A and Open Discussion

Shaheen III Survey