Core Labs and
Research Infrastructure

Programming Environment, Performance & Debugging tools

Bilel Hadri

bilel.hadri@kaust.edu.sa

Computational Scientist

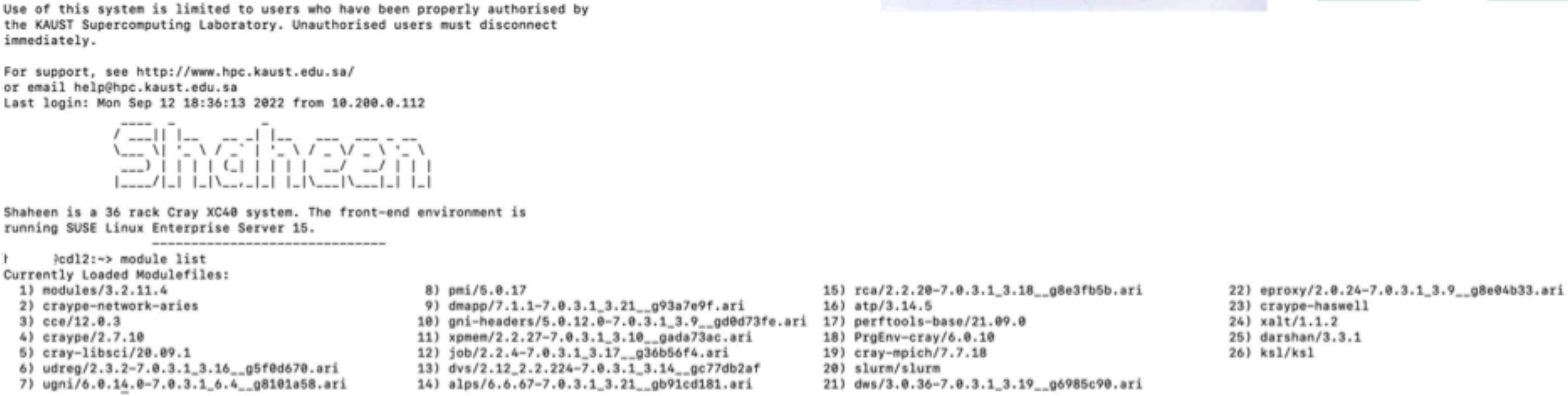
KAUST Supercomputing Lab

@mnoukhiya @KAUST_HPC

SHAHEEN
SUPERCOMPUTING LABORATORY

Shaheen 2 Cray XC40

- Edit and Compile only your code on login. To run, submit jobs.
- **Compiler available Cray CCE (default), Intel and GNU supported**
  - Compiler wrappers for serial and parallel
    - **ftn for Fortran code**
    - **cc for C code**
    - **CC for C++ code**
  - Do not purge.



```
Use of this system is limited to users who have been properly authorised by
the KAUST Supercomputing Laboratory. Unauthorised users must disconnect
immediately.

For support, see http://www.hpc.kaust.edu.sa/
or email help@hpc.kaust.edu.sa
Last login: Mon Sep 12 18:36:13 2022 from 10.200.0.112
                ____  _           _
               / ___|| |__   __ _| |__   ___  ___ _ __
               \___ \| '_ \ / _` | '_ \ / _ \/ _ \ '_ \
                ___) | | | | (_| | | | |  __/  __/ | | |
               |____/|_| |_|\__,_|_| |_|\___|\___|_| |_|

Shaheen is a 36 rack Cray XC40 system. The front-end environment is
running SUSE Linux Enterprise Server 15.
                     --------------------------------
        cdl2:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.4                                 8) pmi/5.0.17                                         15) rca/2.2.20-7.0.3.1_3.18__g8e3fb5b.ari          22) eproxy/2.0.24-7.0.3.1_3.9__g8e04b33.ari
  2) craype-network-aries                             9) dmapp/7.1.1-7.0.3.1_3.21__g93a7e9f.ari             16) atp/3.14.5                                     23) craype-haswell
  3) cce/12.0.3                                      10) gni-headers/5.0.12.0-7.0.3.1_3.9__gd0d73fe.ari     17) perftools-base/21.09.0                         24) xalt/1.1.2
  4) craype/2.7.10                                   11) xpmem/2.2.27-7.0.3.1_3.10__gada73ac.ari            18) PrgEnv-cray/6.0.10                             25) darshan/3.3.1
  5) cray-libsci/20.09.1                             12) job/2.2.4-7.0.3.1_3.17__g36b56f4.ari               19) cray-mpich/7.7.18                              26) ksl/ksl
  6) udreg/2.3.2-7.0.3.1_3.16__g5f0d670.ari          13) dvs/2.12_2.2.224-7.0.3.1_3.14__gc77db2af           20) slurm/slurm
  7) ugni/6.0.14.0-7.0.3.1_6.4__g8101a58.ari         14) alps/6.6.67-7.0.3.1_3.21__gb91cd181.ari            21) dws/3.0.36-7.0.3.1_3.19__g6985c90.ari
```

Compiler Driver Wrappers

Use them exactly like you would use the original compiler, e.g. To compile.

```
#to use Cray  compilers
ftn -o myexe myprog.f90 # Fortran
cc -o myexe myprog.c # for C
CC -o myexe myC++code.C # for C++
```

```
#to use Intel compilers

module swap PrgEnv-cray PrgEnv-intel
ftn -o myexe myprog.f90 # Fortran
cc -o myexe myprog.c # for C
CC -o myexe myC++code.C # for C++
```

```
#to use GNU compilers

module swap PrgEnv-intel PrgEnv-gnu
ftn -o myexe myprog.f90 # Fortran
cc -o myexe myprog.c # for C
CC -o myexe myC++code.C # for C++
```

ftn, cc, and CC, are not Cray compilers; they invoke the Intel, GNU, or Cray compilers under the hood, depending on the loaded programming environment module (PrgEnv-xxx)

No need to call for mpicc/mpif90… only ftn/cc/CC

Compilers

- Intel - better chance of getting processor specific optimizations
- Cray compiler – many new features and optimizations, especially with Fortran; useful tools like reveal work with Cray compiler only
- GNU - widely used by open software

- More information from compilers options on the man page

PrgEnv	Description	Real Compilers
PrgEnv-cray	Cray Compilation Environment	crayftn, craycc, crayCC
PrgEnv-intel	Intel Composer Suite	ifort, icc, icpc
PrgEnv-gnu	GNU Compiler Collection	gfortran, gcc, g++

Core Labs and
Research Infrastructure

Compilers

- Use ftn, cc, and CC to compile instead of the underlying native compilers (ifort, icc, icpc, gfortran, gcc, g++..)
- Use same wrapper even for MPI codes. Do not use mpicc/mpif90….

- Default compiling is dynamic on Shaheen
  - just add the -static flag to the command and link lines,
  - or set CRAYPE_LINK_TYPE=static in the environment

- Compiler wrappers do cross compilation
  - Compiling on login nodes to run on compute nodes
  - One may run into trouble with GNU automake or cmake.
  - Add the specifier –host=x86_64-unknown-linux-gnu for the configure tool .

- By default, Cray C/C++ is using CLANG

OpenMP

OpenMP is supported by all of the PrgEnvs.

PrgEnv	Enable OpenMP
PrgEnv-cray	C/C++: -fopenmp **Fortran: -h omp**
PrgEnv-intel	**-qopenmp**
PrgEnv-gnu	**-fopenmp**

Cray Scientific Libraries

- Compiler wrappers takes care of not only the compiler but also libs like BLAS, LAPACK, SCALAPACK, MPI,..

- Cray Scientific Libraries package, LibSci, is a collection of numerical routines optimized for best performance on Cray systems.
  - LibSci is loaded by default and this is for all programming environment
  - No user flags or options are required for compiling or linking.
  - LibSci library collection contains; BLAS, BLACS, LAPACK, ScaLAPACK, IRT, CRAFFT, CASE, FFT, FFTW2, FFTW3

- Both cray-python and cray-R call the OpenMP threaded version of cray-libscic calls the OpenMP threaded version of cray-libsci.
  - It is recommended to set the number of desired threads with the OMP_NUM_THREADS environment variable.

Cray Scientific Libraries

- FFTW: Cray's main FFT library is FFTW from MIT with some additional optimizations for Cray hardware
- Cray PETSc , Cray Trilinos.
- Just need to module load and compile your code
- No need to put the whole path of the libraries
- Cray-python, cray-R .
  - Just load the module and use the tools

Cray PE DL Plugin

- craype-dl-plugin - introduces the Cray PE DL Plugin for accelerating distributed deep learning DESCRIPTION The Cray PE DL Plugin provides a highly tuned communication layer that can be easily added to any deep learning framework.
- Plugin has both a C and Python 3 API and supports multiple DL datatypes
- Compatible with TensorFlow and PyTorch frameworks
- Can be used with popular DL frameworks or integrated into a project via its API

- module load craype-dl-plugin
- man intro_dl_plugin

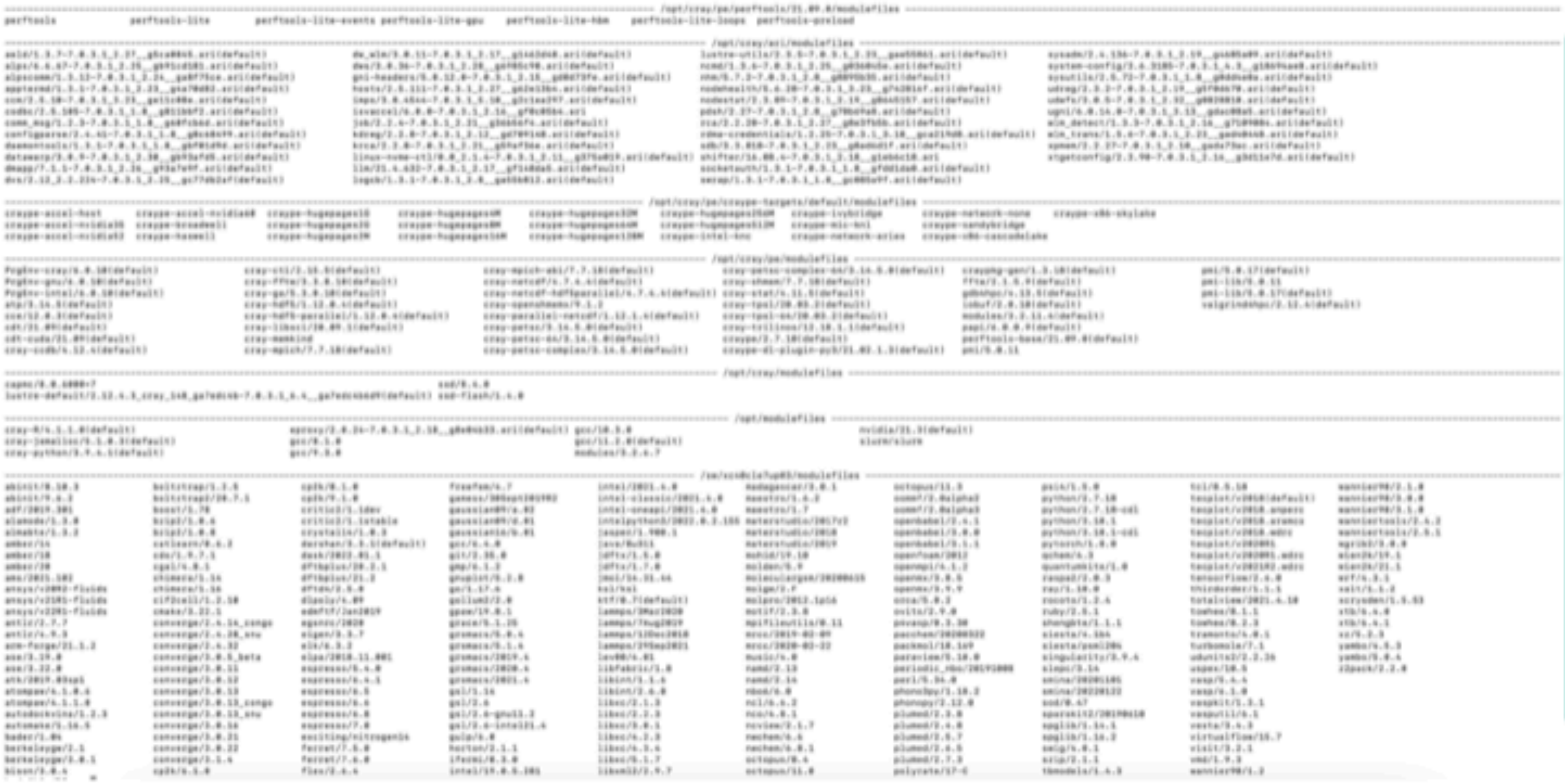
Cray Scientific Libraries

- Cray TPSL (Third Party Scientific Libraries) contains a collection of outside mathematical libraries that can be used with PETSc and Trilinos
  - The TPSL increase the flexibility of PETSc and Trilinos by providing users with multiple options for solving problems in dense and sparse linear algebra
  - The cray-tpsl module is automatically loaded when PETSc or Trilinos is loaded. The libraries included are MUMPs, SuperLU, SuperLU_dist, ParMetis, Hypre, Sundials, and Scotch.

- Intel MKL: The Intel Math Kernel libraries is an alternative to LibSci
  - Features tuned performance for Intel CPUs as well
  - Linking is quite complicated but with Intel compilers (PrgEnv-intel) is usually straightforward using the Intel Link advisor
  - http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

Core Labs and
Research Infrastructure

Modules available on Shaheen



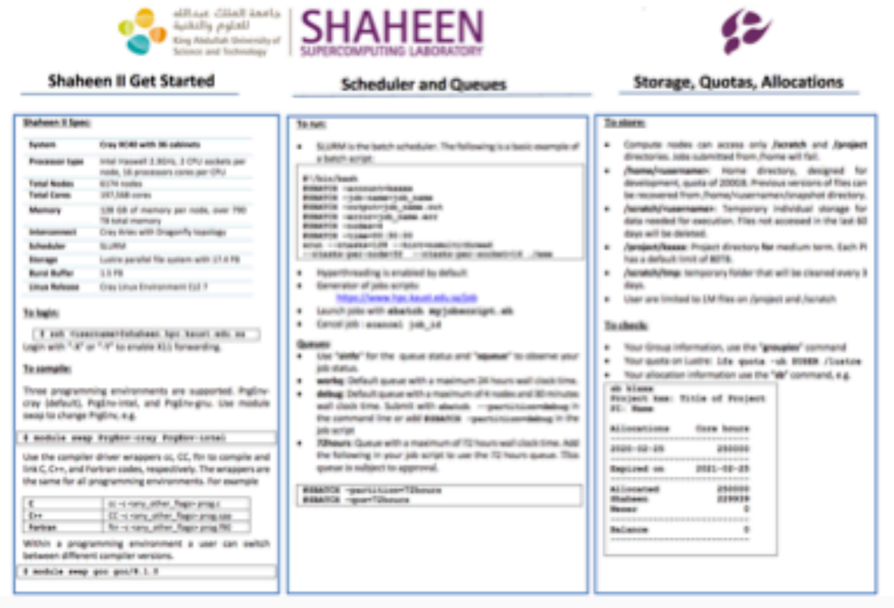
```
------------------------------------------ /opt/cray/pe/perftools/21.09.0/modulefiles ------------------------------------------
perftools          perftools-lite       perftools-lite-events perftools-lite-gpu    perftools-lite-hbm    perftools-lite-loops  perftools-preload

------------------------------------------ /opt/cray/ari/modulefiles ------------------------------------------
[illegible]

------------------------------------------ /opt/cray/pe/craype-targets/default/modulefiles ------------------------------------------
craype-accel-host      craype-accel-nvidia60  craype-hugepages1G     craype-hugepages4M     craype-hugepages32M    craype-hugepages256M   craype-ivybridge       craype-network-none    craype-x86-skylake
craype-accel-nvidia35  craype-broadwell       craype-hugepages2G     craype-hugepages8M     craype-hugepages64M    craype-hugepages512M   craype-mic-knl         craype-sandybridge
craype-accel-nvidia52  craype-haswell         craype-hugepages2M     craype-hugepages16M    craype-hugepages128M   craype-intel-knc       craype-network-aries   craype-x86-cascadelake

------------------------------------------ /opt/cray/pe/modulefiles ------------------------------------------
PrgEnv-cray/6.0.10(default)    cray-cti/[illegible](default)    cray-mpich-abi/7.7.18(default)    cray-petsc-complex-64/3.14.5.0(default)    craypkg-gen/1.3.18(default)    pmi/5.0.17(default)
PrgEnv-gnu/6.0.10(default)     cray-fftw/[illegible](default)   cray-netcdf/[illegible](default)  cray-shmem/7.7.18(default)                 fftw/[illegible](default)      pmi-lib/5.0.11
PrgEnv-intel/6.0.10(default)   cray-ga/[illegible](default)     cray-netcdf-hdf5parallel/[illegible](default)  cray-stat/[illegible](default)  gdb4hpc/[illegible](default)  pmi-lib/5.0.17(default)
atp/[illegible](default)       cray-hdf5/[illegible](default)   cray-openshmemx/[illegible]       cray-tpsl/[illegible](default)             iobuf/[illegible](default)     valgrind4hpc/[illegible](default)
cce/[illegible](default)       cray-hdf5-parallel/[illegible](default)  cray-parallel-netcdf/[illegible](default)  cray-tpsl-64/[illegible](default)  modules/[illegible](default)
cdt/21.09(default)             cray-libsci/[illegible](default) cray-petsc/3.14.5.0(default)      cray-trilinos/[illegible](default)         papi/[illegible](default)
cdt-cuda/21.09(default)        cray-memkind                     cray-petsc-64/3.14.5.0(default)   craype/2.7.10(default)                     perftools-base/21.09.0(default)
cray-ccdb/[illegible](default) cray-mpich/7.7.18(default)       cray-petsc-complex/3.14.5.0(default)  craype-dl-plugin-py3/[illegible](default)  pmi/5.0.11

------------------------------------------ /opt/cray/modulefiles ------------------------------------------
capmc/[illegible]                                          ssd/[illegible]
lustre-default/[illegible](default)                        ssd-flash/[illegible]

------------------------------------------ /opt/modulefiles ------------------------------------------
cray-R/[illegible](default)              aprun/[illegible](default)    gcc/10.3.0                 nvidia/[illegible](default)
cray-jemalloc/[illegible](default)       gcc/8.1.0                     gcc/11.2.0(default)        slurm/slurm
cray-python/[illegible](default)         gcc/9.3.0                     modules/[illegible]

------------------------------------------ /sw/[illegible]/modulefiles ------------------------------------------
abinit/8.10.3      boltztrap/[illegible]   cp2k/8.1.0         freefem/[illegible]   intel/2021.[illegible]   madagascar/[illegible]   [illegible]
abinit/9.[illegible]  boltztrap2/[illegible]  cp2k/9.1.0      gamess/[illegible]    intel-classic/2021.[illegible]  maestro/[illegible]
adf/2019.301       boost/1.78              critic2/[illegible]  gaussian09/[illegible]  intel-oneapi/2021.[illegible]  maestro/1.7
alamode/[illegible]  bzip2/[illegible]     critic2/[illegible]  gaussian09/[illegible]  intelpython3/[illegible]  materstudio/[illegible]
[illegible]
```

Core Labs and
Research Infrastructure

Shaheen 2 Cray XC40 flyer

- **Get started with the flyer**

https://hpc.kaust.link/Shaheen_flyer



SHAHEEN
SUPERCOMPUTING LABORATORY

Shaheen II Get Started

Shaheen II Spec:

System	**Cray XC40 with 36 cabinets**
Processor type	Intel Haswell 2.3GHz, 2 CPU sockets per node, 16 processors cores per CPU
Total Nodes	6174 nodes
Total Cores	197,568 cores
Memory	128 GB of memory per node, over 790 TB total memory
Interconnect	Cray Aries with Dragonfly topology
Scheduler	SLURM
Storage	Lustre parallel file system with 17.6 PB
Burst Buffer	1.5 PB
Linux Release	Cray Linux Environment CLE 7

To login:

```
$ ssh <username>@shaheen.hpc.kaust.edu.sa
```

Login with "-X" or "-Y" to enable X11 forwarding.

To compile:

Three programming environments are supported: PrgEnv-cray (default), PrgEnv-intel, and PrgEnv-gnu. Use module swap to change PrgEnv, e.g.

```
$ module swap PrgEnv-cray PrgEnv-intel
```

Use the compiler driver wrappers cc, CC, ftn to compile and link C, C++, and Fortran codes, respectively. The wrappers are the same for all programming environments. For example

C	cc -o <any_other_flags> prog.c
C++	CC -o <any_other_flags> prog.cpp
Fortran	ftn -o <any_other_flags> prog.f90

Within a programming environment a user can switch between different compiler versions.

```
$ module swap gcc gcc/8.1.0
```

Scheduler and Queues

To run:

- SLURM is the batch scheduler. The following is a basic example of a batch script:

```
#!/bin/bash
#SBATCH --account=kxxxx
#SBATCH --job-name=job_name
#SBATCH --output=job_name.out
#SBATCH --error=job_name.err
#SBATCH --nodes=4
#SBATCH --time=01:00:00
srun --ntasks=128 --hint=nomultithread
--ntasks-per-node=32 --ntasks-per-socket=16 ./exe
```

- Hyperthreading is enabled by default.
- Generator of jobs scripts:
  https://www.hpc.kaust.edu.sa/job
- Launch jobs with **sbatch my_jobscript.sh**
- Cancel job : **scancel job_id**

Queues:

- Use "**sinfo**" for the queue status and "**squeue**" to observe your job status.
- **workq**: Default queue with a maximum 24 hours wall clock time.
- **debug**: Default queue with a maximum of 4 nodes and 30 minutes wall clock time. Submit with **sbatch --partition=debug** in the command line or add **#SBATCH --partition=debug** in the job script.
- **72hours**: Queue with a maximum of 72 hours wall clock time. Add the following in your job script to use the 72 hours queue. This queue is subject to approval.

```
#SBATCH --partition=72hours
#SBATCH --qos=72hours
```

Storage, Quotas, Allocations

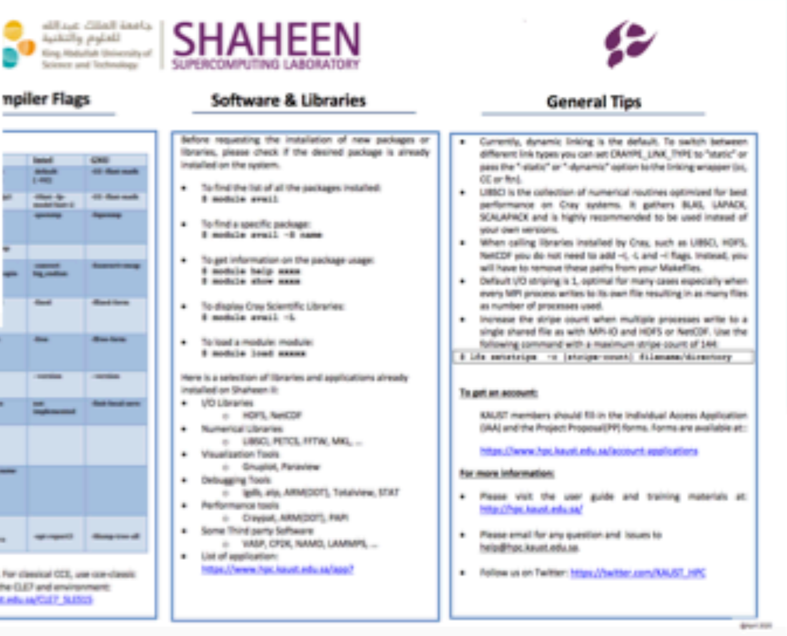
To store:

- Compute nodes can access only **/scratch** and **/project** directories. Jobs submitted from /home will fail.
- **/home/<username>**: Home directory, designed for development, quota of 200GB. Previous versions of files can be recovered from /home/<username>/snapshot directory.
- **/scratch/<username>**: Temporary individual storage for data needed for execution. Files not accessed in the last 60 days will be deleted.
- **/project/kxxxx**: Project directory **for** medium term. Each PI has a default limit of 80TB.
- **/scratch/tmp**: temporary folder that will be cleaned every 3 days.
- User are limited to 1M files on /project and /scratch

To check:

- Your Group information, use the "**groups**" command
- Your quota on Lustre: **lfs quota -uh $USER /lustre**
- Your allocation information use the "**sb**" command, e.g.

```
sb k1xxx
Project kxxx: Title of Project
PI: Name

Allocations        Core hours
------------------------------
2020-02-25             250000
------------------------------
Expired on         2021-02-25
------------------------------
Allocated              250000
Shaheen                229939
Neser                       0
------------------------------
Balance                     0
------------------------------
```



SHAHEEN
SUPERCOMPUTING LABORATORY

mpiler Flags

		Intel	GNU
[illegible]	[illegible]	default (-O2)	-O2 -ffast-math
[illegible]	[illegible]	-fast -fp-model fast=2	-O3 -ffast-math
[illegible]	[illegible]	-qopenmp	-fopenmp
[illegible]	[illegible]		
[illegible]	[illegible]	-convert big_endian	-fconvert=swap
[illegible]	[illegible]	-fixed	-ffixed-form
Process Fortran source using free form specifications	-f free	-free	-ffree-form
Show version number of the compiler	-V	--version	--version
Zero fill all uninitialized variables	-e zero	not implemented	-finit-local-zero
Creates .mod files to hold Fortran90 module information for future compiles	-e m		
Specifies the directory to which .file.mod files are written when the -e m option is specified	-J dir_name		
Getting compiler feedback, produces .lst files	-hlist=a	-opt-report3	-fdump-tree-all

- CCE is based on LLVM. For classical CCE, use cce-classic
- More information on the CLE7 and environment:
  https://www.hpc.kaust.edu.sa/CLE7_SLES15

Software & Libraries

Before requesting the installation of new packages or libraries, please check if the desired package is already installed on the system.

- 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 xxxx`
  `$ module show xxxx`
- To display Cray Scientific Libraries:
  `$ module avail -L`
- To load a module: module:
  `$ module load xxxxx`

Here is a selection of libraries and applications already installed on Shaheen II:

- I/O Libraries
  - HDF5, NetCDF
- Numerical Libraries
  - LIBSCI, PETCS, FFTW, MKL, ...
- Visualization Tools
  - Gnuplot, Paraview
- Debugging Tools
  - lgdb, atp, ARM(DDT), Totalview, STAT
- Performance tools
  - Craypat, ARM(DDT), PAPI
- Some Third party Software
  - VASP, CP2K, NAMD, LAMMPS, ...
- List of application:
  https://www.hpc.kaust.edu.sa/app7

General Tips

- Currently, dynamic linking is the default. To switch between different link types you can set CRAYPE_LINK_TYPE to "static" or pass the "-static" or "-dynamic" option to the linking wrapper (cc, CC or ftn).
- LIBSCI is the collection of numerical routines optimized for best performance on Cray systems. It gathers BLAS, LAPACK, SCALAPACK and is highly recommended to be used instead of your own versions.
- When calling libraries installed by Cray, such as LIBSCI, HDF5, NetCDF you do not need to add –I, -L and –l flags. Instead, you will have to remove these paths from your Makefiles.
- Default I/O striping is 1, optimal for many cases especially when every MPI process writes to its own file resulting in as many files as number of processes used.
- Increase the stripe count when multiple processes write to a single shared file as with MPI-IO and HDF5 or NetCDF. Use the following command with a maximum stripe count of 144.

```
$ lfs setstripe  -c [stripe-count] filename/directory
```

To get an account:

KAUST members should fill in the Individual Access Application (IAA) and the Project Proposal(PP) forms. Forms are available at:

https://www.hpc.kaust.edu.sa/account-applications

For more information:

- Please visit the user guide and training materials at: http://hpc.kaust.edu.sa/
- Please email for any question and issues to help@hpc.kaust.edu.sa
- Follow us on Twitter: https://twitter.com/KAUST_HPC

CDT: updates

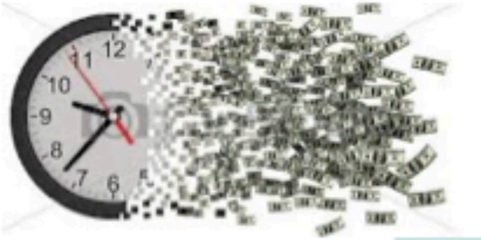
- In order, KSL will be updating the Cray Programming Environment, namely the Cray Developer Toolkit (CDT), to provide a predictable, stable and consistent programming environment while still making necessary software updates.
- Using more recent packages may result in faster execution of the code.
- CDT consists of compilers, MPI, scientific and I/O libraries, profiling and debugging tools, etc.
- New CDT software will be installed at least twice a year. The new versions will not be made the defaults when installed. You need to load them.
- Module load cdt/21.09 is the default.
  - cdt/22.09 is available

Performance

Why Performance Analysis ?

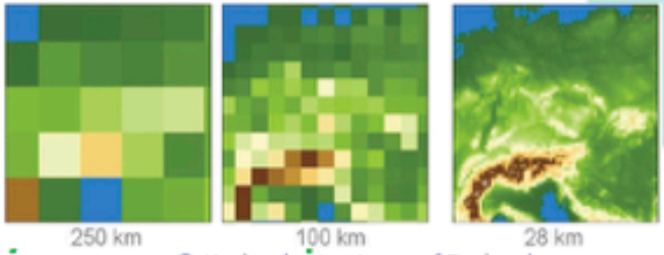
- You want to get the best expected performance.
  - Ex: Internet Bandwidth, RPM vehicles
  - Need to identity the issue

- Economic: TIME is MONEY
  - Lifetime of HPC systems is short (4/5 years)
  - Large HPC machines cost in O($10M)



- Qualitative: Do more science
  - Get codes run faster
  - Perform more time steps
  - Simulation higher resolutions

- **Must strive to evaluate how your code is running.**
- **Learn to think of performance during the entire cycle of**



© Karlsruhe Institute of Technology

Core Labs and Research Infrastructure

Typical Performance Analysis Procedure

- Measuring the wallclock time is not enough.
- Need to know what's really happening under the hood.

- Do I have a performance problem at all?
  - Time / speedup / scalability
- What is the key bottleneck ?
  - computation / communication
- Where is the key bottleneck?
  - Detailed profiling
- Why does the code have scalability problems?
  - Load imbalance analysis, compare profiles at various sizes function-by -function, performance modeling

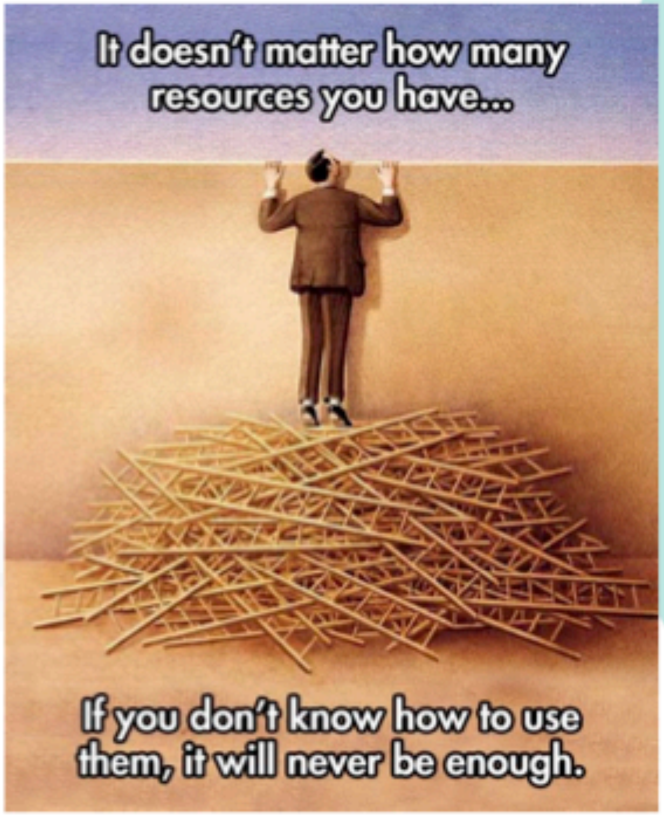
Performance measurement

- No single solution is sufficient
  - Timing manually… Not efficient and accurate
  - Don’t reinvent the wheel

- Need to use a combination of different methods, tools and techniques is needed!
  - Measurement Sampling and profiling
  - Analysis Statistics, visualization, automatic analysis, data mining, ...

Performance/Monitoring tools

- Many tools are available on HPC systems:
  - Gprof
  - PAPI
  - VTUNE
  - Allinea/ARM Tools
  - VAMPIR
  - TAU
  - Scalasca
  - Likwid
  - VAMPIR
  - HPCToolkit
  - Paraver/Extrae
  - Darshan
  - Perftools (Cray systems)

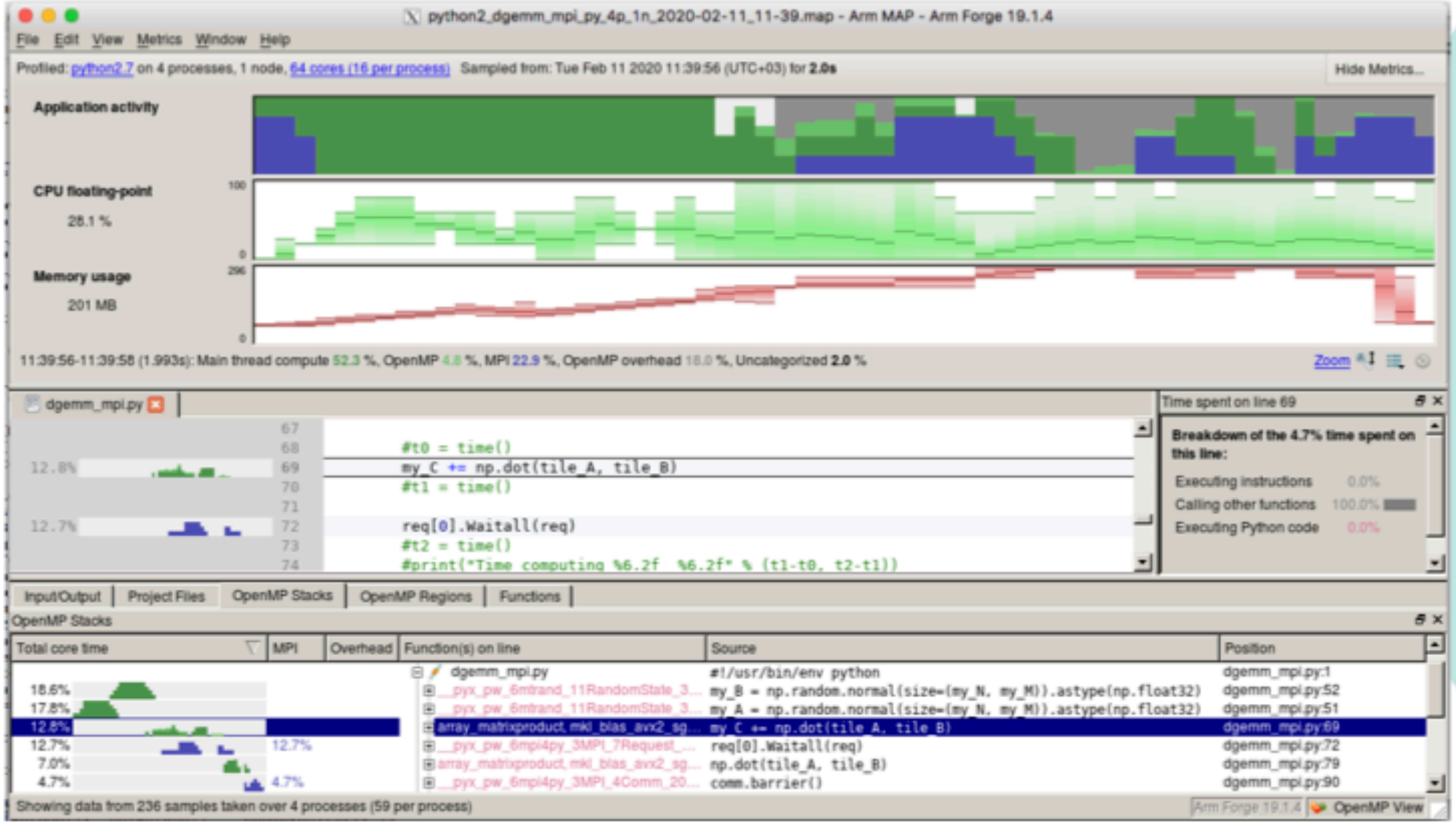

It doesn't matter how many resources you have...
If you don't know how to use them, it will never be enough.

Profile a Python code

- Just type:
  - python -m cProfile myscript.py

- For call graphs
  - pycallgraph graphviz -- ./myscript.py
  - Display pycallgraph.png

Overview on performance of code over time



python2_dgemm_mpi_py_4p_1n_2020-02-11_11-39.map - Arm MAP - Arm Forge 19.1.4

Profiled: python2.7 on 4 processes, 1 node, 64 cores (16 per process) Sampled from: Tue Feb 11 2020 11:39:56 (UTC+03) for **2.0s**

11:39:56-11:39:58 (1.993s): Main thread compute 52.3 %, OpenMP 4.8 %, MPI 22.9 %, OpenMP overhead 18.0 %, Uncategorized **2.0** %

```
67
68          #t0 = time()
69          my_C += np.dot(tile_A, tile_B)
70          #t1 = time()
71
72          req[0].Waitall(req)
73          #t2 = time()
74          #print("Time computing %6.2f  %6.2f" % (t1-t0, t2-t1))
```

Time spent on line 69

Breakdown of the 4.7% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%
Executing Python code	0.0%

OpenMP Stacks

Total core time	MPI	Overhead	Function(s) on line	Source	Position
			⊟ dgemm_mpi.py	`#!/usr/bin/env python`	dgemm_mpi.py:1
18.6%			⊞ __pyx_pw_6mtrand_11RandomState_3...	`my_B = np.random.normal(size=(my_N, my_M)).astype(np.float32)`	dgemm_mpi.py:52
17.8%			⊞ __pyx_pw_6mtrand_11RandomState_3...	`my_A = np.random.normal(size=(my_N, my_M)).astype(np.float32)`	dgemm_mpi.py:51
12.8%			⊞ array_matrixproduct, mkl_blas_avx2_sg...	`my_C += np.dot(tile_A, tile_B)`	dgemm_mpi.py:69
12.7%	12.7%		⊞ __pyx_pw_6mpi4py_3MPI_7Request_...	`req[0].Waitall(req)`	dgemm_mpi.py:72
7.0%			⊞ array_matrixproduct, mkl_blas_avx2_sg...	`np.dot(tile_A, tile_B)`	dgemm_mpi.py:79
4.7%	4.7%		⊞ __pyx_pw_6mpi4py_3MPI_4Comm_20...	`comm.barrier()`	dgemm_mpi.py:90

Showing data from 236 samples taken over 4 processes (59 per process)

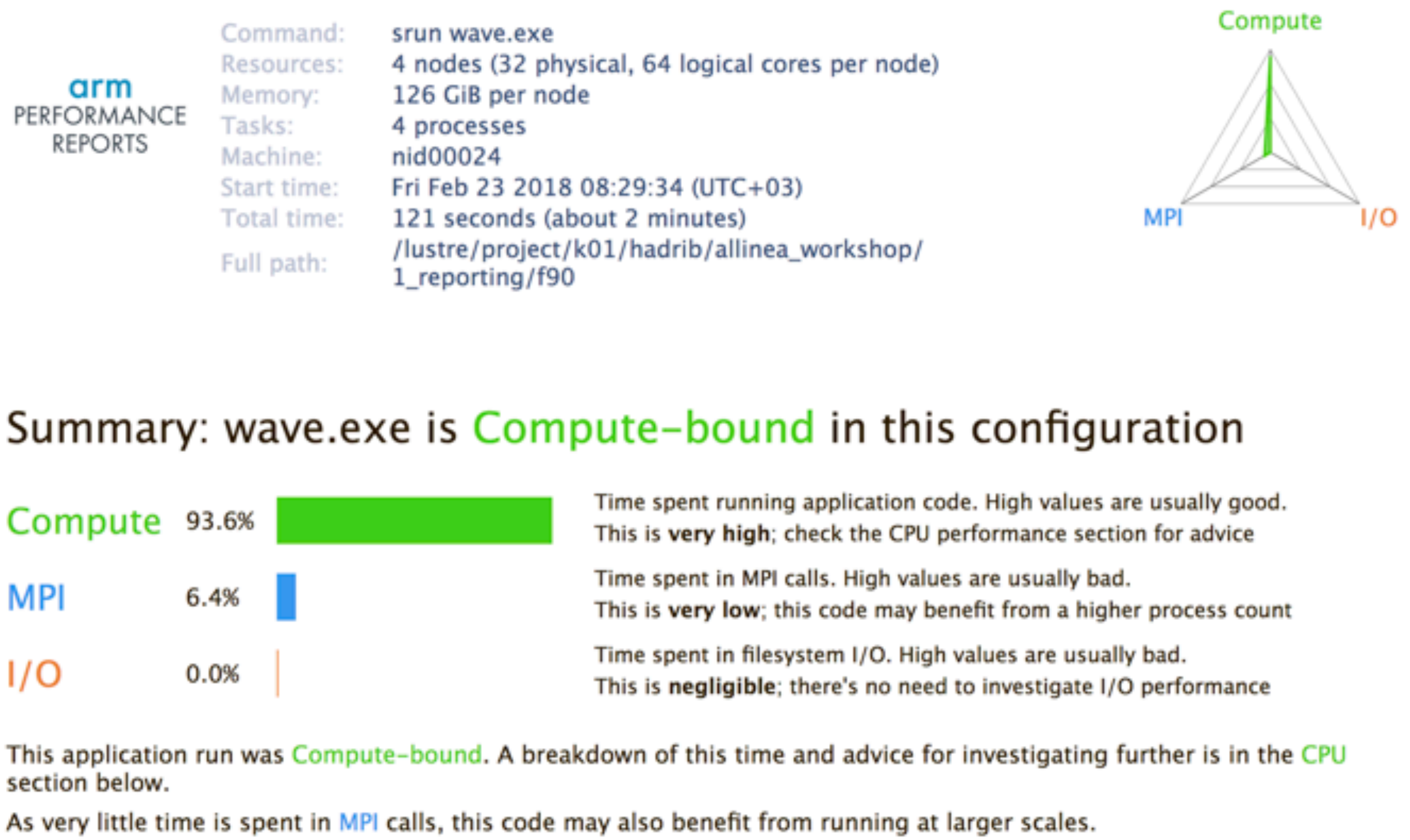
ARM/DDT

- ARM performance Tools

- Provides quick overview of performance issues:
  - The time spent in various categories of instruction: memory access, numeric operations, floating point operations
  - Overview on I/O, Memory, Communication, Threads , Energy usage
  - Energy Saves data in HTML, CVS or text form

- To get the report in html or txt
  - Load arm-reports module
  - make-profiler-libraries
  - Relink dynamically your code as shown in the output
  - perf-report srun –n 2 ./mycode

ARM/DDT general Overview



arm PERFORMANCE REPORTS

Command: srun wave.exe
Resources: 4 nodes (32 physical, 64 logical cores per node)
Memory: 126 GiB per node
Tasks: 4 processes
Machine: nid00024
Start time: Fri Feb 23 2018 08:29:34 (UTC+03)
Total time: 121 seconds (about 2 minutes)
Full path: /lustre/project/k01/hadrib/allinea_workshop/1_reporting/f90

Summary: wave.exe is Compute-bound in this configuration

Compute 93.6% — Time spent running application code. High values are usually good. This is **very high**; check the CPU performance section for advice

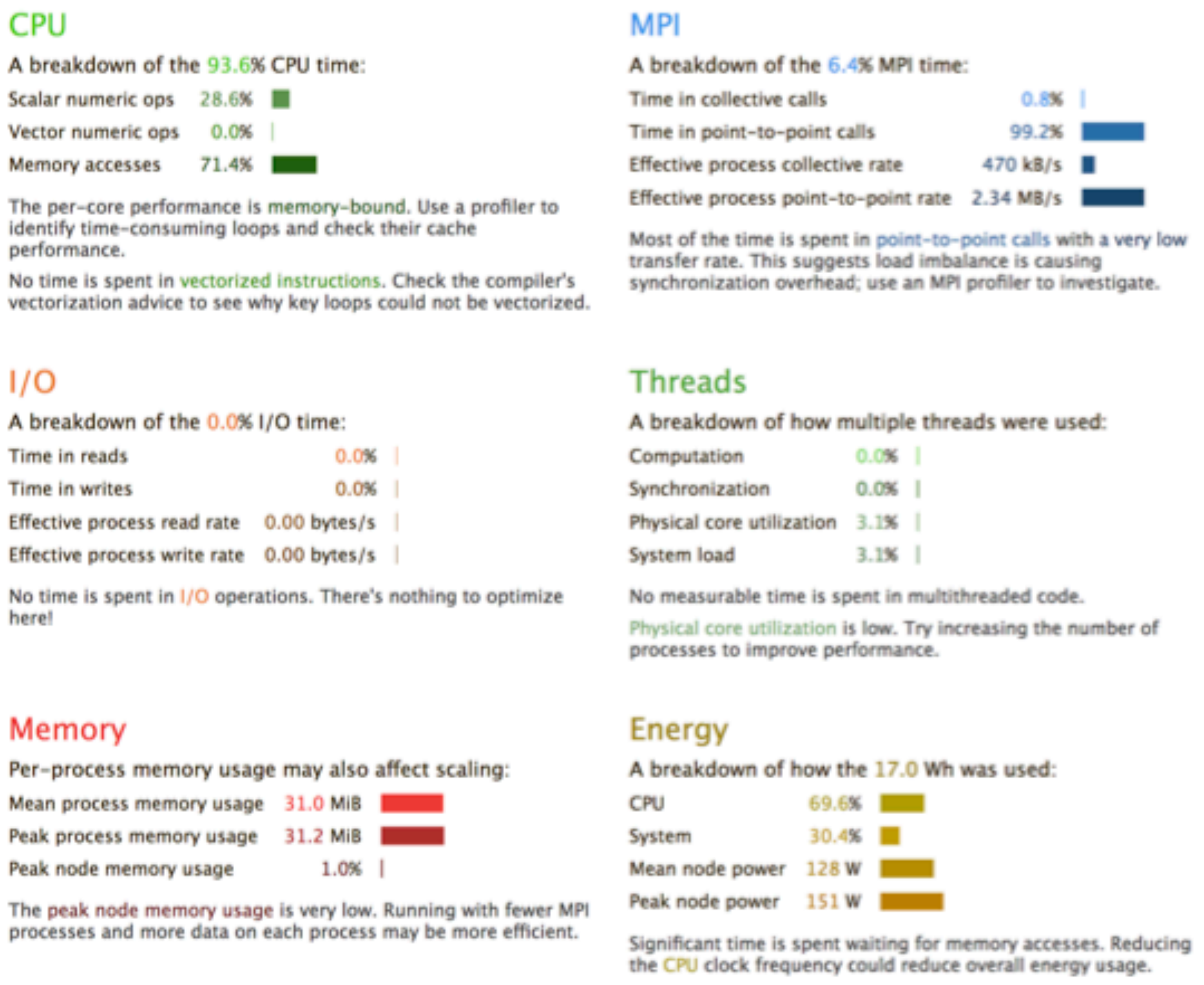
MPI 6.4% — Time spent in MPI calls. High values are usually bad. This is **very low**; this code may benefit from a higher process count

I/O 0.0% — Time spent in filesystem I/O. High values are usually bad. This is **negligible**; there's no need to investigate I/O performance

This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

As very little time is spent in MPI calls, this code may also benefit from running at larger scales.

ARM/DDT Detailed



CPU

A breakdown of the 93.6% CPU time:

Scalar numeric ops	28.6%
Vector numeric ops	0.0%
Memory accesses	71.4%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

No time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

MPI

A breakdown of the 6.4% MPI time:

Time in collective calls	0.8%
Time in point-to-point calls	99.2%
Effective process collective rate	470 kB/s
Effective process point-to-point rate	2.34 MB/s

Most of the time is spent in point-to-point calls with a very low transfer rate. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

I/O

A breakdown of the 0.0% I/O time:

Time in reads	0.0%
Time in writes	0.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	0.00 bytes/s

No time is spent in I/O operations. There's nothing to optimize here!

Threads

A breakdown of how multiple threads were used:

Computation	0.0%
Synchronization	0.0%
Physical core utilization	3.1%
System load	3.1%

No measurable time is spent in multithreaded code.

Physical core utilization is low. Try increasing the number of processes to improve performance.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	31.0 MiB
Peak process memory usage	31.2 MiB
Peak node memory usage	1.0%

The peak node memory usage is very low. Running with fewer MPI processes and more data on each process may be more efficient.

Energy

A breakdown of how the 17.0 Wh was used:

CPU	69.6%
System	30.4%
Mean node power	128 W
Peak node power	151 W

Significant time is spent waiting for memory accesses. Reducing the CPU clock frequency could reduce overall energy usage.

Core Labs and
Research Infrastructure

How is my IO ?

- Use profiling and charactirization tools
  - Allinea report,
  - Craypat profiling
  - Darshan
  - Contact CS team at KSL

Summary: benchio_allinea is I/O-bound in this configuration

Compute

MPI

I/O

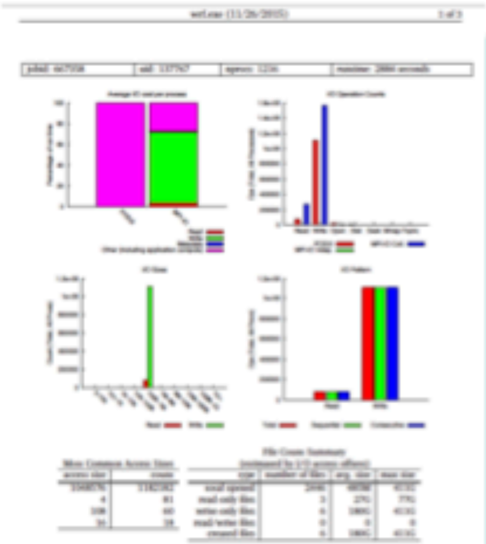
CPU

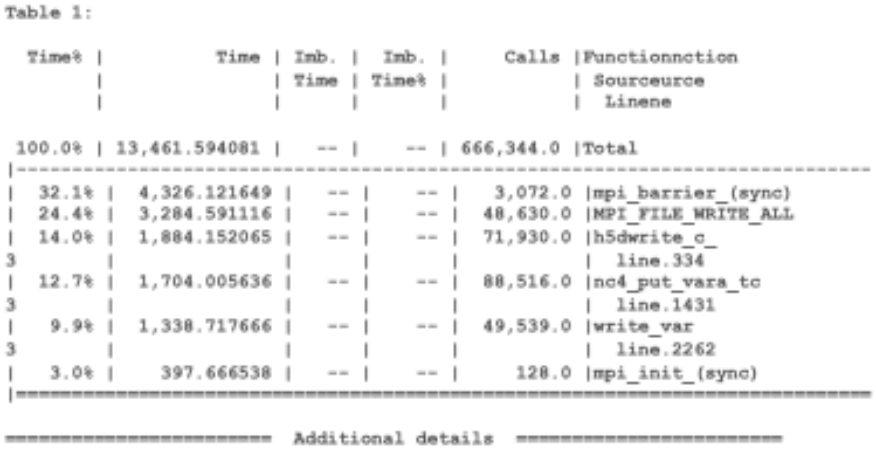
MPI

I/O

Threads

Memory





```
Table 1:

  Time%  |           Time  |  Imb.  |   Imb.  |      Calls  |Functionnction
         |                 |  Time  |  Time%  |             | Sourceurce
         |                 |        |         |             |   Linene

 100.0%  | 13,461.594081  |    --  |     --  | 666,344.0  |Total
|-------------------------------------------------------------------------
|   32.1% |  4,326.121649 |    -- |     -- |    3,072.0 |mpi_barrier_(sync)
|   24.4% |  3,284.591116 |    -- |     -- |   48,630.0 |MPI_FILE_WRITE_ALL
|   14.0% |  1,884.152065 |    -- |     -- |   71,930.0 |h5dwrite_c_
3         |               |       |        |            |  line.334
|   12.7% |  1,704.005636 |    -- |     -- |   88,516.0 |nc4_put_vara_tc
3         |               |       |        |            |  line.1431
|    9.9% |  1,338.717666 |    -- |     -- |   49,539.0 |write_var
3         |               |       |        |            |  line.2262
|    3.0% |    397.666538 |    -- |     -- |      128.0 |mpi_init_(sync)
|=========================================================================

=======================  Additional details  =======================
```

Debugging

Core Labs and
Research Infrastructure

Valgrind4hpc

- Valgrind4hpc debugging tool helps in the detection of memory leaks and errors in parallel applications.
- **Compile and link with -g option** , then allocate and follow the steps shown bellow.

```
salloc -N 1 module

unload darshan xalt

module load valgrind4hpc

export CTI_WLM_IMPL=slurm export CTI_LAUNCHER_NAME=srun

valgrind4hpc -n2 --launcher-args="--hint=nomultithread --ntasks=2" --valgrind-args="--track-
origins=yes --leak-check=full" ./my_exe
```

- Here is a clean output. Otherwise, follow the instructions to detect the memory leaks:

```
RANKS: <0,1>

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

All heap blocks were freed -- no leaks are possible ERROR SUMMARY: 0 errors from 0 contexts
(suppressed 19)
```

- To run your program and debug it across multiple nodes, allocate the desired number of nodes and then update accordingly the parameters in the launcher-args similar to the option for the srun/sbatch script.
- More information is available in the man pages of valgrind and valgrind4hpc.

Different tools available

- Several tools for C/C++/Fortran debugging tools:
  - gdb4hpc
  - valgrind4hpc
  - sanitize4hpc

gdb4hpc

- gdb4hpc (Cray Line Mode Parallel Debugger) is a GDB-based parallel debugger, developed by Cray.
- It can debug with CCE, PGI, GNU and Intel Fortran, C and C++ compilers.

- **gdb4hpc** also includes comparative debugger technology that enables programmers to compare data structures between two executing applications. Cray, however, recommends accessing the comparative debugger technology through the new Cray Comparative Debugger (CCDB) with graphical user interface (GUI) that enhances the parallel debugging capabilities of **gdb4hpc**.

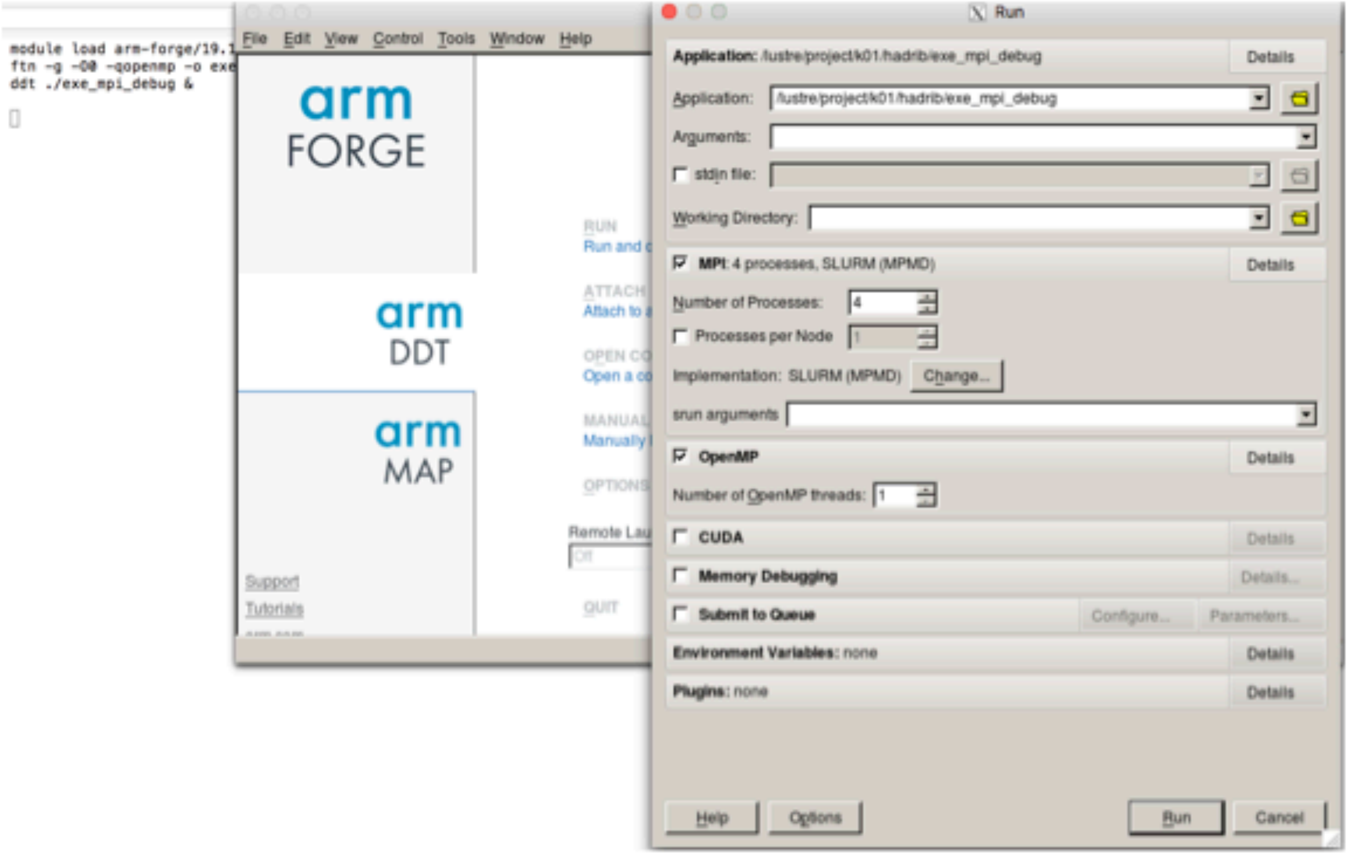
- More info in man pages

```
module load gdb4hpc
```

Note: need to unload xalt

Debugging with ARM/DDT

```
> ftn -g -O0 -qopenmp -o exe_mpi_debug test_hib.f90
> salloc -N 1
> module load arm-forge/19.1.4
> ddt exe_mpi_debug
```



```
module load arm-forge/19.1
ftn -g -O0 -qopenmp -o exe
ddt ./exe_mpi_debug &
```

Application: /lustre/project/k01/hadrib/exe_mpi_debug

Application: /lustre/project/k01/hadrib/exe_mpi_debug

Arguments:

□ stdin file:

Working Directory:

☑ MPI: 4 processes, SLURM (MPMD)

Number of Processes: 4

□ Processes per Node 1

Implementation: SLURM (MPMD)

srun arguments

☑ OpenMP

Number of OpenMP threads: 1

□ CUDA

□ Memory Debugging

□ Submit to Queue

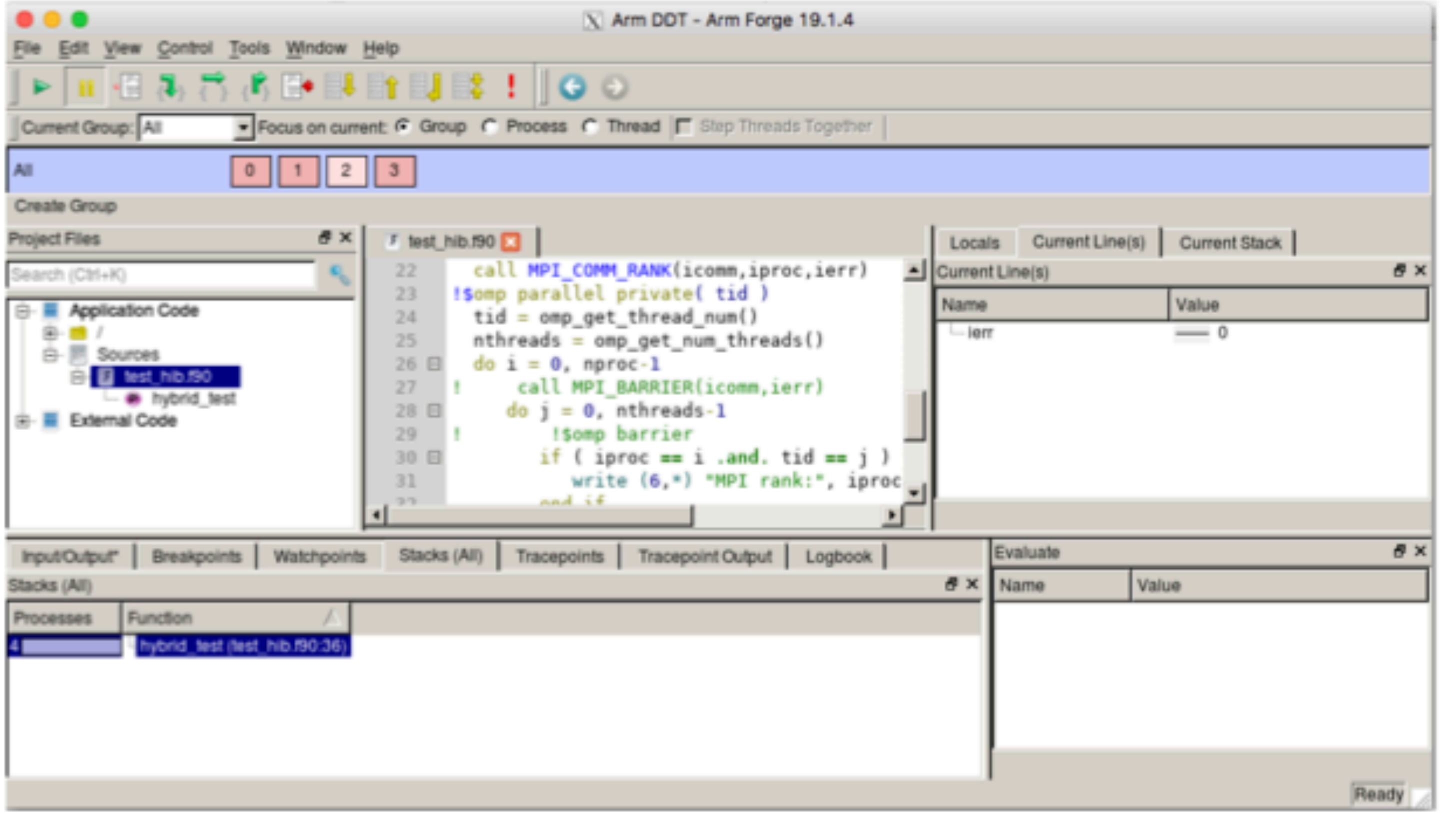
Environment Variables: none

Plugins: none

Core Labs and Research Infrastructure

Debugging with ARM/DDT

Use the GUI to navigate within the code and check the variable



```
  call MPI_COMM_RANK(icomm,iproc,ierr)
!$omp parallel private( tid )
  tid = omp_get_thread_num()
  nthreads = omp_get_num_threads()
  do i = 0, nproc-1
!       call MPI_BARRIER(icomm,ierr)
     do j = 0, nthreads-1
!          !$omp barrier
        if ( iproc == i .and. tid == j )
           write (6,*) "MPI rank:", iproc
        end if
```

Current Line(s)

Name	Value
ierr	0

Stacks (All)

Processes	Function
4	hybrid_test (test_hib.f90:36)

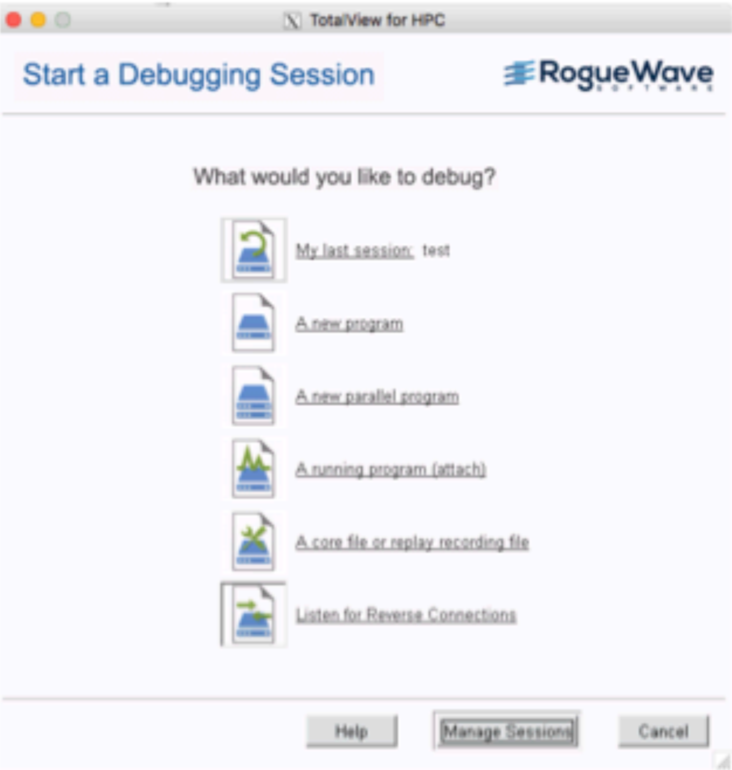
Evaluate

Name	Value

Debugging with Totalview

- Compile with –g option as usual
- Allocate the node

```
>module load totalview
>tv8 &
```



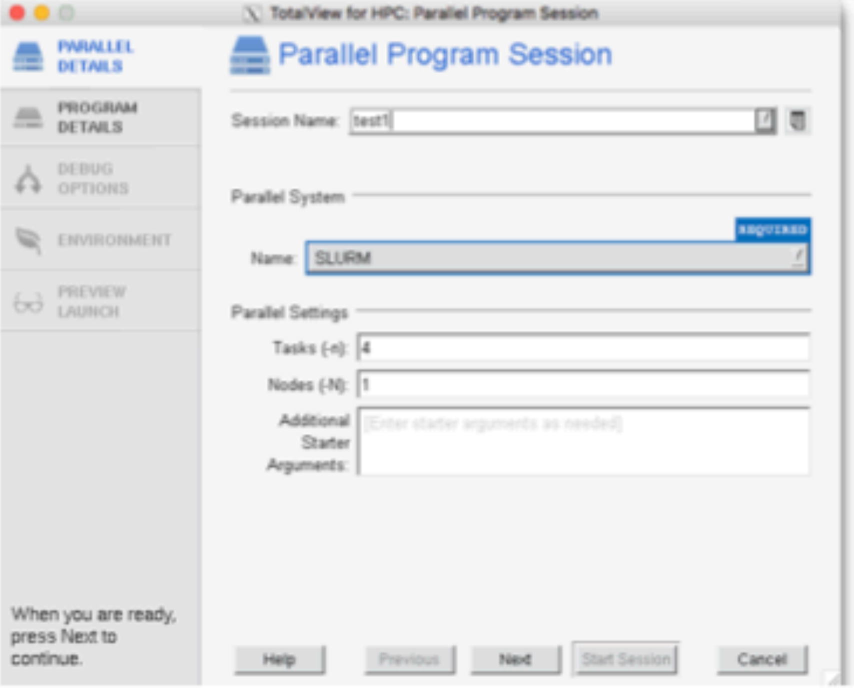
TotalView for HPC

Start a Debugging Session

What would you like to debug?

My last session: test

RogueWave
SOFTWARE
A new program
A new parallel program
A running program (attach)
A core file or replay recording file
Listen for Reverse Connections
Help
Manage Sessions
Cancel



TotalView for HPC: Parallel Program Session

Parallel Program Session

Session Name: test1

Parallel System

Name: SLURM

Parallel Settings

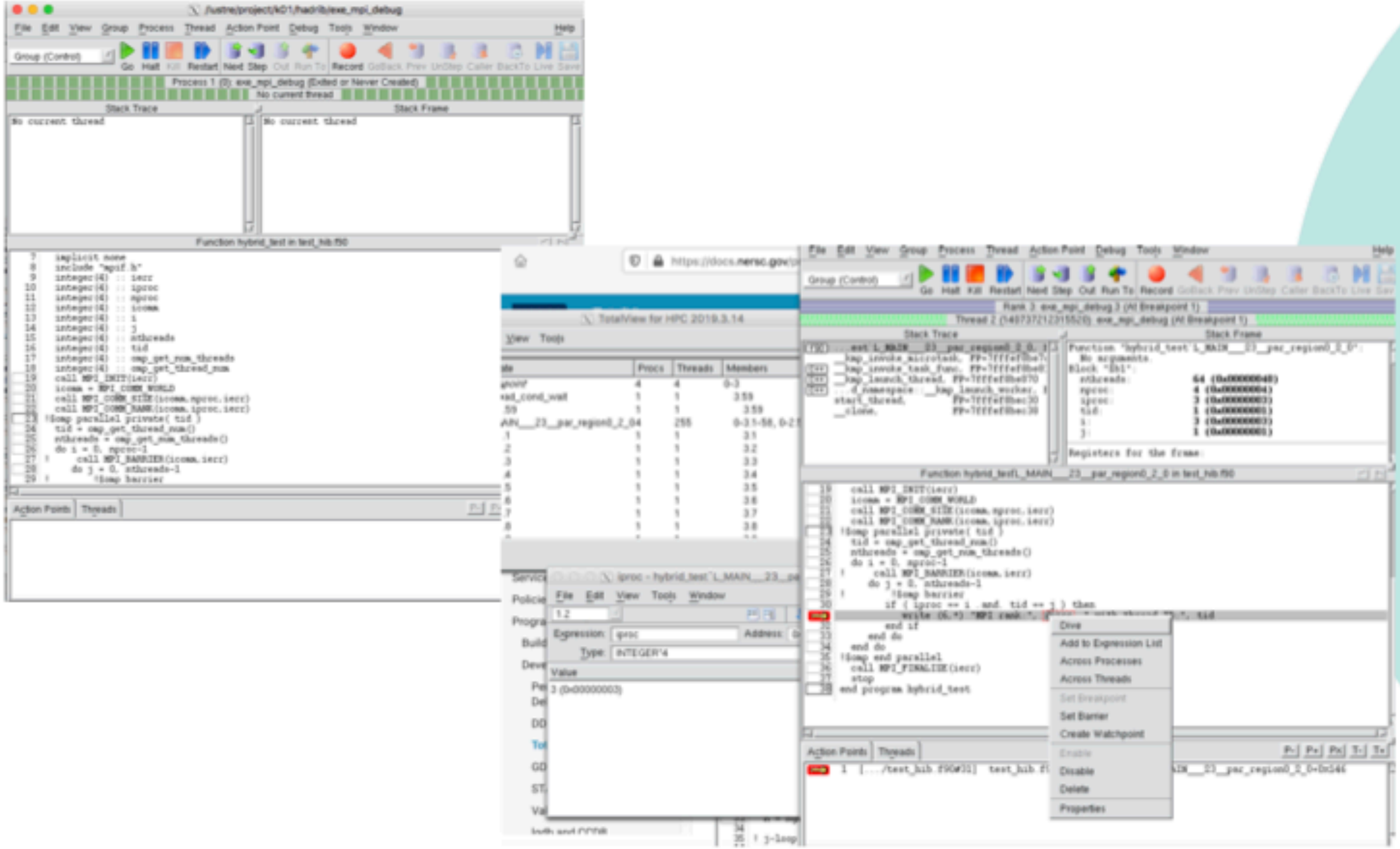
Tasks (-n): 4

Nodes (-N): 1

Additional Starter Arguments: [Enter starter arguments as needed]

When you are ready, press Next to continue.

Totalview


/lustre/project/k01/hadrib/exe_mpi_debug
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live Save
Stack Trace
Stack Frame
Action Points Threads

Process 1 (0): exe_mpi_debug (Exited or Never Created)

No current thread

```
No current thread
```

```
No current thread
```

Function hybrid_test in test_hib.f90

```
  implicit none
  include "mpif.h"
  integer(4) :: ierr
  integer(4) :: iproc
  integer(4) :: nproc
  integer(4) :: icomm
  integer(4) :: i
  integer(4) :: j
  integer(4) :: nthreads
  integer(4) :: tid
  integer(4) :: omp_get_num_threads
  integer(4) :: omp_get_thread_num
  call MPI_INIT(ierr)
  icomm = MPI_COMM_WORLD
  call MPI_COMM_SIZE(icomm,nproc,ierr)
  call MPI_COMM_RANK(icomm,iproc,ierr)
!$omp parallel private( tid )
  tid = omp_get_thread_num()
  nthreads = omp_get_num_threads()
  do i = 0, nproc-1
!     call MPI_BARRIER(icomm,ierr)
     do j = 0, nthreads-1
!       !$omp barrier
```

https://docs.nersc.gov/

TotalView for HPC 2019.3.14

...de	Procs	Threads	Members
...point/	4	4	0-3
...ead_cond_wait	1	1	3.59
.59	1	1	3.59
...AIN__23__par_region0_2_04		255	0-3.1-58, 0-2.1
.1	1	1	3.1
.2	1	1	3.2
.3	1	1	3.3
.4	1	1	3.4
.5	1	1	3.5
.6	1	1	3.6
.7	1	1	3.7
.8	1	1	3.8

iproc - hybrid_test`L_MAIN__23__pa

1.2

Expression: iproc

Address:

Type: INTEGER*4

Value

3 (0x00000003)

File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live Save
Stack Trace
Stack Frame

Rank 3: exe_mpi_debug.3 (At Breakpoint 1)

Thread 2 (140737212315520): exe_mpi_debug (At Breakpoint 1)

```
F90  ...est`L_MAIN__23__par_region0_2_0, F
     __kmp_invoke_microtask,  FP=7fffef0be7
C++  __kmp_invoke_task_func,  FP=7fffef0be8
C++  __kmp_launch_thread,     FP=7fffef0be870
C++  ...d_namespace::__kmp_launch_worker, F
     start_thread,            FP=7fffef0bec30
     __clone,                 FP=7fffef0bec38
```

```
Function "hybrid_test`L_MAIN__23__par_region0_2_0":
  No arguments.
Block "$b1":
  nthreads:               64 (0x00000040)
  nproc:                  4 (0x00000004)
  iproc:                  3 (0x00000003)
  tid:                    1 (0x00000001)
  i:                      3 (0x00000003)
  j:                      1 (0x00000001)

Registers for the frame:
```

Function hybrid_test`L_MAIN__23__par_region0_2_0 in test_hib.f90

```
  call MPI_INIT(ierr)
  icomm = MPI_COMM_WORLD
  call MPI_COMM_SIZE(icomm,nproc,ierr)
  call MPI_COMM_RANK(icomm,iproc,ierr)
!$omp parallel private( tid )
  tid = omp_get_thread_num()
  nthreads = omp_get_num_threads()
  do i = 0, nproc-1
!     call MPI_BARRIER(icomm,ierr)
     do j = 0, nthreads-1
!       !$omp barrier
        if ( iproc == i .and. tid == j ) then
           write (6,*) "MPI rank ", iproc, " with thread ", tid
        end if
     end do
  end do
!$omp end parallel
  call MPI_FINALIZE(ierr)
  stop
end program hybrid_test
```

```
1  [.../test_hib.f90#31]  test_hib.f9 ...AIN__23__par_region0_2_0+0x146
```

Transferring Files

Core Labs and
Research Infrastructure

Transfer files: Use Globus

- scp/ftp for small size (in order of KB)

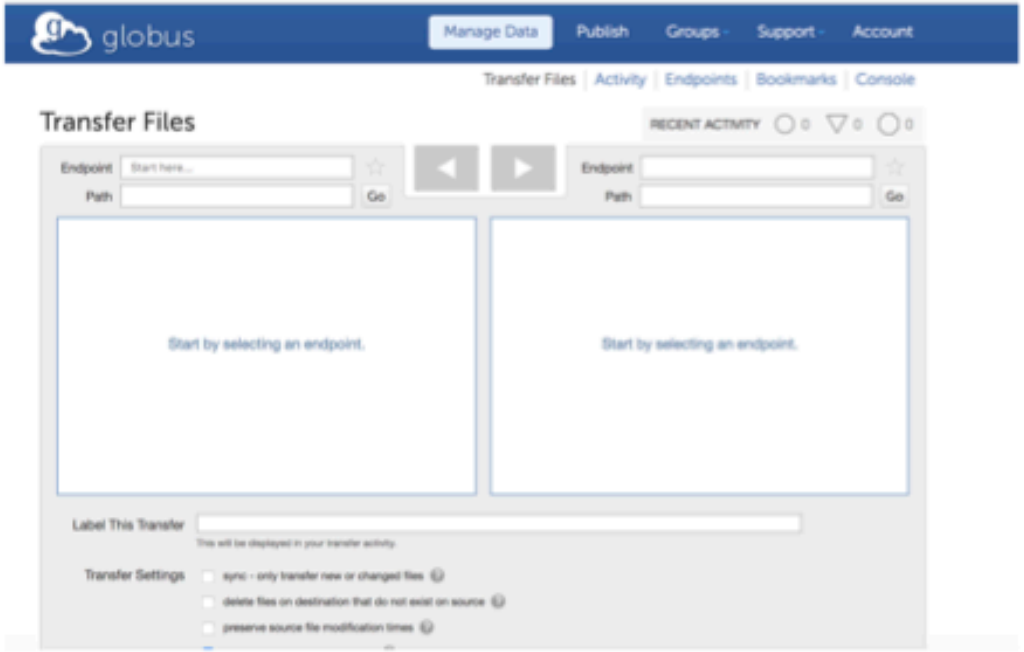
- For larger file, use Globus, especially for moving data in & out of Shaheen http://www.globus.org/ (Free)
  - Reliable & easy-to-use web-based service:
  - Email notification of success or failure

- Globus extensive documentation https://docs.globus.org
    - Web based interaction with service
    - REST/API for scripted interactions with service
    - Globus Connect Server & Personal for setting up additional remote endpoints such your personal laptop/ workstation

- Globus on Shaheen. Look for Shaheen End point Point
  - Within Campus: choose dm2.hpc.kaust.edu.sa
  - Outside Campus: choose dm1.hpc.kaust.edu.sa

Globus

- Connect to globus.org
- Sign in or create an account
- Use Shaheen dm2 when inside KAUST and dm1 when connected externally.



Transfer Files

Start by selecting an endpoint.

Start by selecting an endpoint.

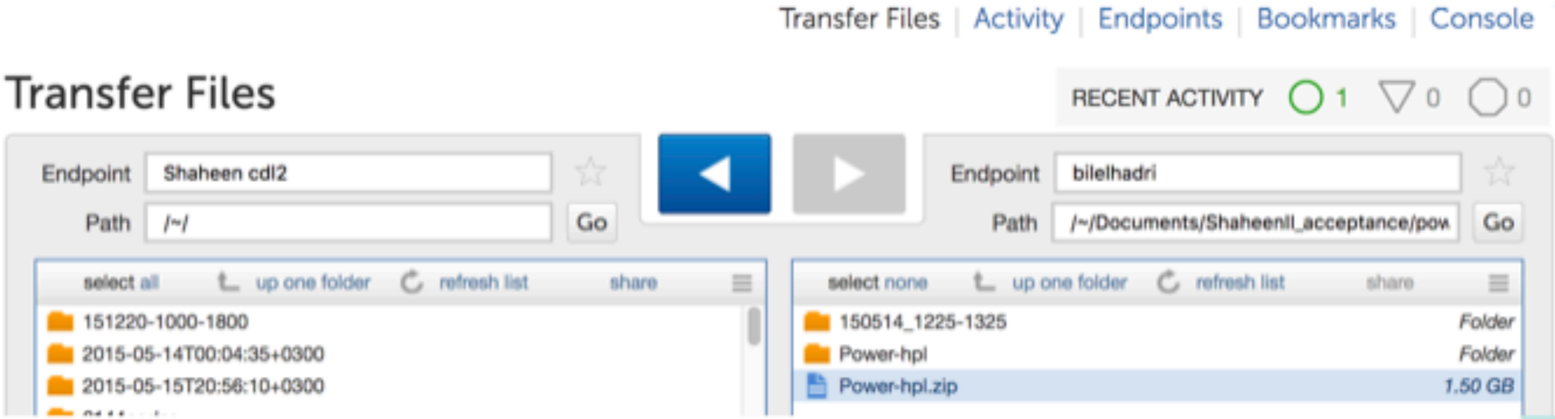
Label This Transfer

This will be displayed in your transfer activity.

Transfer Settings
- □ sync - only transfer new or changed files
- □ delete files on destination that do not exist on source
- □ preserve source file modification times

Core Labs and
Research Infrastructure

Transferring with Globus


Transfer Files | Activity | Endpoints | Bookmarks | Console
RECENT ACTIVITY ○ 1 ▽ 0 ⬡ 0
select all
up one folder
refresh list
share
select none
Go
Go

Transfer Files

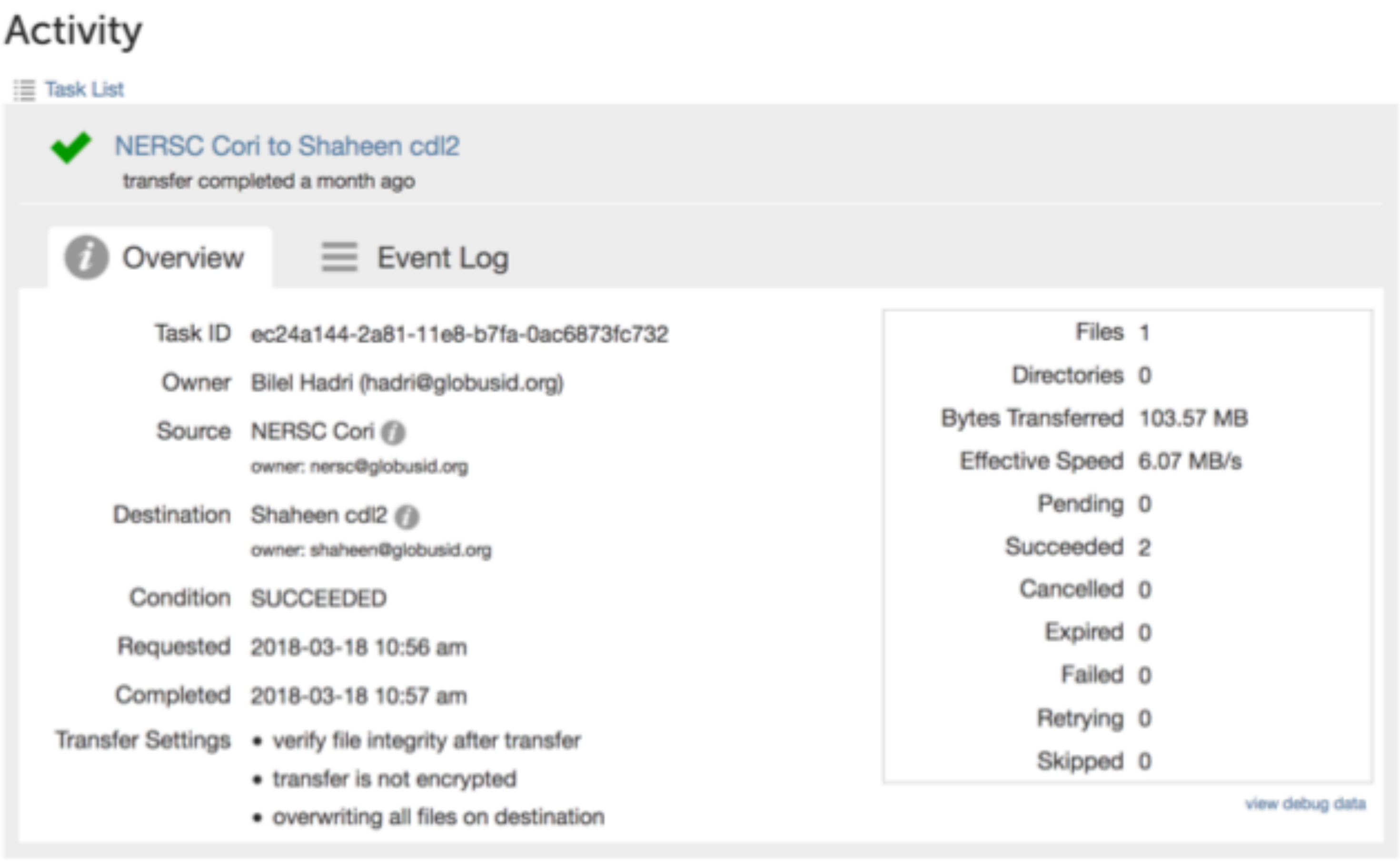
Endpoint: Shaheen cdl2
Path: /~/

- 151220-1000-1800
- 2015-05-14T00:04:35+0300
- 2015-05-15T20:56:10+0300

Endpoint: bilelhadri
Path: /~/Documents/ShaheenII_acceptance/pow

150514_1225-1325	Folder
Power-hpl	Folder
Power-hpl.zip	1.50 GB

File Status notification (email and web-interface)



Activity

Task List

✓ NERSC Cori to Shaheen cdl2

transfer completed a month ago

Overview | Event Log

Task ID	ec24a144-2a81-11e8-b7fa-0ac6873fc732
Owner	Bilel Hadri (hadri@globusid.org)
Source	NERSC Cori owner: nersc@globusid.org
Destination	Shaheen cdl2 owner: shaheen@globusid.org
Condition	SUCCEEDED
Requested	2018-03-18 10:56 am
Completed	2018-03-18 10:57 am
Transfer Settings	• verify file integrity after transfer • transfer is not encrypted • overwriting all files on destination

Files	1
Directories	0
Bytes Transferred	103.57 MB
Effective Speed	6.07 MB/s
Pending	0
Succeeded	2
Cancelled	0
Expired	0
Failed	0
Retrying	0
Skipped	0

view debug data

Tips & Summary

Best Practices for Performance (1)

- Check the system details thoroughly
  - Never assume ! (Login nodes different than compute)

- Choose a compiler and MPI to build your application
  - All are not same ! Rely on the latest versions
- Start with some basic compiler flags and try additional flags one at a time
  - Optimization is incremental ! Benchmarking and testing is a must
- Use the built-in/optimized libraries and tools to save time and improve performance
  - Libraries Tools are your friends !
  - By doing the different steps of optimizations:
  - You can achieve huge speedup (o(10x) and more) by using Optimized Mathematics libraries (Cray, MKL)
  - Optimizing the cache and memory

Core Labs and Research Infrastructure

Best Practices for Performance (2)

- Don’t Reinvent the wheel ! Several tools are available for debugging and performance

- Test your application at every level to arrive at an optimized code
  - Check correctness !
- Customize your runtime environment to achieve desired goals
  - Play with the number of threads, memory and core affinity
- Profile and adjust optimization and runtime environments accordingly
  - Start with small and short runs
- **READ the manual and/or attend the tutorials/workshops !**
- **Visit https://www.hpc.kaust.edu.sa/training**

Best Practices

- Use adequately your allocation
  - Check your core hours, sb kxxxx ,sb_user kxxxx
  - Check your quota usage kuq, kpq
  - Prepare in advance the project proposal
- Shaheen is a shared resource
  - Be kind to your neighbor users
  - Don’t run on login.
- Follow the terms and conditions
  - Don’t share your account with others.

KSL CS Team

- Need help: send a ticket help@hpc.kaust.edu.sa
  - Help us to help you :D
    - Provide details.
    - Which HPC system?
    - What is the problem? When did it happen? What modules were loaded? How did you try to fix or work around it? Send the error and job script.

- **Acknowledge KAUST Supercomputing Lab and HPC resources used in your papers.**

Core Labs and
Research Infrastructure

bilel.hadri@kaust.edu.sa

@mnoukhiya   @KAUST_HPC

Thank You !   شكراً !