Shaheen Cray Programming Environment, Performance and IO overview

Bilel Hadri

bilel.hadri@kaust.edu.sa
Computational Scientist
KAUST Supercomputing Lab

@mnoukhiya @KAUST_HPC

SHAHEEN SUPERCOMPUTING LABORATORY
Outline

1. Shaheen Programming Environment
2. Performance
3. IO Filesystem
4. Neser
5. Tips
Shaheen Programming Environment
Shaheen 2 Cray XC40

- Edit and Compile only your code on login. TO run, submit jobs
- Cray CCE by default, Intel and GNU supported
  - Just module swap to PrgEnv-intel or PrgEnv-gnu
  - Compiler wrappers for serial and parallel
    - ftn for Fortran code
    - cc for C code
    - CC for C++ code
  - Do not purge.

```
Bilel Hadri – Shaheen Workshop Jan 2019
```
Compiler Driver Wrappers

- Use them exactly like you would use the original compiler, e.g. To compile `prog.f90` run

  ```
  ftn -o myprog.exe myprog.f90
  ```

- These scripts choose which compiler to use from the loaded PrgEnv module

<table>
<thead>
<tr>
<th>PrgEnv</th>
<th>Description</th>
<th>Real Compilers</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgEnv-crty</td>
<td>Cray Compilation Environment</td>
<td>crayftn, craycc, crayCC</td>
</tr>
<tr>
<td>PrgEnv-intel</td>
<td>Intel Composer Suite</td>
<td>ifort, icc, icpc</td>
</tr>
<tr>
<td>PrgEnv-gnu</td>
<td>GNU Compiler Collection</td>
<td>gfortran, gcc, g++</td>
</tr>
</tbody>
</table>
OpenMP

- OpenMP is supported by all of the PrgEnvs.
  - CCE (PrgEnv-cray) recognizes and interprets OpenMP directives by default. If you have OpenMP directives in your application but do not want to use them, disable OpenMP recognition with \(-hnoomp\).

<table>
<thead>
<tr>
<th>PrgEnv</th>
<th>Enable OpenMP</th>
<th>Disable OpenMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgEnv-cray</td>
<td>-homp</td>
<td>-hnoomp</td>
</tr>
<tr>
<td>PrgEnv-intel</td>
<td>-qopenmp</td>
<td></td>
</tr>
<tr>
<td>PrgEnv-gnu</td>
<td>-fopenmp</td>
<td></td>
</tr>
</tbody>
</table>

-\(qopenmp\) is the replacement option for -openmp, which is deprecated.
• Use ftn, cc, and CC to compile instead of the underlying native compilers (ifort, icc, icpc, gfortran, gcc, g++..)T
  – Compiler wrappers wraps the underlying compilers with the additional compiler and linker flags depending on the modules loaded in the environment

• Compilers wrappers link statically by default
  – Recommended for performance at scale

• Set environment variable CRAYPE_LINK_TYPE=dynamic to link dynamically
  – may take a some time to load shared libraries at runtime

• 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.
Cray Scientific Libraries

- Compiler wrappers take care of not only the compiler but also libraries like BLAS, SCALAPACK, MPI, etc.

- 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 environments.
  - 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

- FFTW: Cray’s main FFT library is FFTW from MIT with some additional optimizations for Cray hardware.

- Cray PETSc (with CASK – Cray Adaptive Sparse Kernels)

- Cray Trilinos (with CASK – Cray Adaptive Sparse Kernels)

- Just need to module load and compile your code
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
Modules available on Shaheen

Bilel Hadri – Shaheen Workshop Jan 2019
Modules snapshot

• Avoid editing the .bashrc. Use module snapshot

• The basic idea is that you take a snapshot of your environment and save it with a name. You can then restore it whenever you need. You can have many snapshot “files” saved too.

• So, if for example you work with two (or more) applications that require different environment setup, you can have a snapshot for each one of them, and whenever you need to work on one of them, you just restore the corresponding snapshot.

• The usage is:
  – module snapshot [-f | --force] <filename>
  – module restore <filename>
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.

CDT consists of compilers, MPI, scientific and I/O libraries, profiling and debugging tools, etc.

New CDT software will be installed regularly.

cdt/17.12 cdt/18.09 available as of today

cdt/19.02 will be available in February 2019.

To get the desired PrgEnv cdt, just swap the default to the desired

Check and use the latest version for faster
Check the flyer

• Available on line: http://tiny.cc/KSL_flyer
Performance
Why Performance Analysis?

- You want to get the best expected performance.
  - Ex: Internet Bandwidth, RPM vehicles
  - Need to identify 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 your code development.
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)
Profile a Python code

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

- For call graphs
  - `pycallgraph graphviz -- ./myscript.py`
  - Display `pycallgraph.png`
• Allinea now 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

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

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

<table>
<thead>
<tr>
<th>Compute</th>
<th>93.6%</th>
<th><strong>Time spent running application code. High values are usually good. This is very high; check the CPU performance section for advice</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>6.4%</td>
<td><strong>Time spent in MPI calls. High values are usually bad. This is very low; this code may benefit from a higher process count</strong></td>
</tr>
<tr>
<td>I/O</td>
<td>0.0%</td>
<td><strong>Time spent in filesystem I/O. High values are usually bad. This is negligible; there's no need to investigate I/O performance</strong></td>
</tr>
</tbody>
</table>

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.

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!

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.

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.

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.

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.
• Cray-KAUST tutorials:
  – How to use perftools
  – How to use reveal for OpenMP parallelization
Overview on I/O
Hardware File System

- Shaheen Cray XC40
  - 6174 nodes of 32 cores Haswell with a total of 792 TB of memory
  - Cray Sonexion® 2000 Storage System with 17.2 PB of usable capacity with performance exceeding 500 GB/sec.
  - Cray DataWarp with a capacity of 1.5 PB and a performance exceeding 1.5 TB/sec
  - Cray Tiered Adaptive Storage (TAS) with 30 PB of capacity (up to 100 PB)
## Data Storage

<table>
<thead>
<tr>
<th>Area</th>
<th>Path</th>
<th>Type</th>
<th>Quota</th>
<th>Backups</th>
<th>Purged</th>
</tr>
</thead>
<tbody>
<tr>
<td>User home</td>
<td>/home/username</td>
<td>NFS</td>
<td>200 G</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>User project</td>
<td>/project/kxxx</td>
<td>Lustre</td>
<td>80TB per PI</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>User scratch</td>
<td>/scratch/username</td>
<td>Lustre</td>
<td>-</td>
<td>No</td>
<td>Yes 60 days</td>
</tr>
<tr>
<td>Scratch project</td>
<td>/scratch/kxxx</td>
<td>Lustre</td>
<td>-</td>
<td>No</td>
<td>Yes 60 days</td>
</tr>
<tr>
<td>Scratch tmp</td>
<td>/scratch/tmp</td>
<td>Lustre</td>
<td>-</td>
<td>No</td>
<td>Yes 3 days</td>
</tr>
</tbody>
</table>

- Home directory, designed for development. Previous versions of files can be recovered from `/home/<username>/`.snapshot directory.
HPC systems and I/O

- "A supercomputer is a device for converting a CPU-bound problem into an I/O bound problem." [Ken Batcher]

- **Machines consist of three main components:**
  - Compute nodes
  - High-speed interconnect
  - I/O infrastructure

- **Most optimization work on HPC applications is carried out on**
  - **Single node performance**
  - Network performance (communication)
  - I/O only when it becomes a real problem
How is my IO?

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

Table 1:

| Time% | Time | Imb. | Imb. | Calls | Function
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>13,461.594081</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>Linene</td>
</tr>
<tr>
<td>32.1%</td>
<td>4,326.121649</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>mpi_barrier_(sync)</td>
</tr>
<tr>
<td>24.4%</td>
<td>3,284.591116</td>
<td>--</td>
<td>--</td>
<td>48,630.0</td>
<td>MPI_FILE_WRITE_ALL</td>
</tr>
<tr>
<td>14.0%</td>
<td>1,884.152065</td>
<td>--</td>
<td>--</td>
<td>71,930.0</td>
<td>h5dwrite_c_</td>
</tr>
<tr>
<td>12.7%</td>
<td>1,704.005636</td>
<td>--</td>
<td>--</td>
<td>88,516.0</td>
<td>nc4_put_vara_tc</td>
</tr>
<tr>
<td>9.9%</td>
<td>1,338.717666</td>
<td>--</td>
<td>--</td>
<td>49,539.0</td>
<td>write_var</td>
</tr>
<tr>
<td>3.0%</td>
<td>397.666538</td>
<td>--</td>
<td>--</td>
<td>128.0</td>
<td>mpi_init_(sync)</td>
</tr>
</tbody>
</table>

Additional details
Darshan

- Darshan is “a scalable HPC I/O characterization tool... designed to capture an accurate picture of application I/O behavior... with minimum overhead”
  - I/O Characterization
  - Sheds light on the intricacies of an application’s I/O
  - Useful for application I/O debugging, Pinpointing causes of extremes
  - Analyzing/tuning hardware for optimizations
- Installed by default on Shaheen
  - Requires no code modification (only re-linking)
  - Includes a job summary tool:
    - Location of the $DARSHAN_LOGPATH/YYYY/MM/DD/username_exe_jobid_xxx.gz
    - module load texlive/2017
    - darshan-job-summary.pl username_exe_jobid_xxx.darshan
    - Open the new PDF file
Darshan Report on IOR benchmark

ior (2/24/2018) 1 of 3

I/O performance estimate (at the POSIX layer): transferred 2330 MB at 447288.07 MB/s
I/O performance estimate (at the STDIO layer): transferred 0.0 MB at 22.84 MB/s

File Count Summary

<table>
<thead>
<tr>
<th>Access size</th>
<th>Count</th>
<th>type</th>
<th>number of files</th>
<th>avg. size</th>
<th>max size</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX</td>
<td>1648576</td>
<td>total opened</td>
<td>2305</td>
<td>12G</td>
<td>57G</td>
</tr>
<tr>
<td></td>
<td></td>
<td>read-only files</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>write-only files</td>
<td>2305</td>
<td>12G</td>
<td>57G</td>
</tr>
<tr>
<td></td>
<td></td>
<td>read/write files</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>created files</td>
<td>2305</td>
<td>12G</td>
<td>57G</td>
</tr>
</tbody>
</table>

Average I/O per process (POSIX and STDERR)

<table>
<thead>
<tr>
<th>Type</th>
<th>Cumulative time spent in I/O functions (seconds)</th>
<th>Amount of I/O (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent reads</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Independent writes</td>
<td>61.1901351158854</td>
<td>12211.5520842978</td>
</tr>
<tr>
<td>Independent metadata</td>
<td>0.170816845052083</td>
<td>N/A</td>
</tr>
<tr>
<td>Shared reads</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shared writes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shared metadata</td>
<td>0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Data Transfer Per Filesystem (POSIX and STDERR)

<table>
<thead>
<tr>
<th>File System</th>
<th>Write MB</th>
<th>Write Ratio</th>
<th>Read MB</th>
<th>Read Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>28135416.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>0.00222</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Bilel Hadri – Shaheen Workshop Jan 2019
Optimizations

• Do not reinvent the wheel
• You can optimize your code/application with no change
  – file striping to increase IO performance:
    • set the adequate stripe size in your directory
  – IOBUF for serial I/O operations
    • Just load a module
• Use I/O libraries: HDF5, NetCDF, ADIOS
  – Need refactoring your code, however you will further get performance and productivity for file processing using state of the art visualization tools
Lustre filestripping

• Files on the Lustre filesystems can be striped
  – Transparently divided into chunks that are written or read simultaneously across a set of OSTs within the filesystem.
  – The chunks are distributed among the OSTs using a method that ensures load balancing.

• Benefits include:
  – Allows one or more clients to read/write different parts of the same file at the same time,
  – Providing higher I/O bandwidth to the file because the bandwidth is aggregated over the multiple OSTs.

• Large files must be striped in order to avoid taking up too much space on any single OST, which might adversely affect the filesystem.
Useful Lustre commands

- Listing Striping Information
  - lfs getstripe filename
  - lfs getstripe -d directory_name

- File stripping:
  - lfs setstripe -s stripe_size -c stripe_count dir/filename

- Note: The stripe settings of an existing file cannot be changed. If you want to change the settings of a file, create a new file with the desired settings and copy the existing file to the newly created file.
<table>
<thead>
<tr>
<th>Stripe count</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>time I/O</td>
<td>79</td>
<td>48</td>
<td>37</td>
<td>42</td>
<td>39</td>
</tr>
<tr>
<td>time code</td>
<td>122</td>
<td>91</td>
<td>83</td>
<td>87</td>
<td>85</td>
</tr>
<tr>
<td>%I/O</td>
<td>65%</td>
<td>53%</td>
<td>45%</td>
<td>48%</td>
<td>46%</td>
</tr>
<tr>
<td>Speedup IO</td>
<td>1.00</td>
<td>1.65</td>
<td>2.14</td>
<td>1.88</td>
<td>2.03</td>
</tr>
<tr>
<td>Speedup code</td>
<td>1.00</td>
<td>1.34</td>
<td>1.47</td>
<td>1.40</td>
<td>1.44</td>
</tr>
</tbody>
</table>
WRF 12x speedup with file striping

Default striping 1,
I/O time: 2094 sec
Total time: 2884 sec

<table>
<thead>
<tr>
<th>File Count Summary</th>
<th>(estimated by I/O access offsets)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type</td>
</tr>
<tr>
<td>total opened</td>
<td>2446</td>
</tr>
<tr>
<td>read-only files</td>
<td>3</td>
</tr>
<tr>
<td>write-only files</td>
<td>6</td>
</tr>
<tr>
<td>read/write files</td>
<td>0</td>
</tr>
<tr>
<td>created files</td>
<td>6</td>
</tr>
</tbody>
</table>

Stripping over 144 I/O
time: 174 sec
Total time: 959 sec

I/O speedup: 12x
Total time speedup: 3x
Check my usage

• **Command:** `lfs quota -hu username /lustre/`

Disk quotas for user xxxx (uid 1.....5):

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>used</th>
<th>quota</th>
<th>limit</th>
<th>grace</th>
<th>files</th>
<th>quota</th>
<th>limit</th>
<th>grace</th>
</tr>
</thead>
<tbody>
<tr>
<td>/lustre/</td>
<td>3.62T</td>
<td>0k</td>
<td>0k</td>
<td>-</td>
<td>181413</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• **Recommendation:**
  - Keep only needed data and below the limits
    - Limit is 1M of files by default
    - Error message:
      - **UNRECOVERABLE error on system request** Disk quota exceeded
  - Purge will be applied on /scratch
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.
This page is deprecated and will be discontinued April 2019. Please Use the New Version and update your links.
Transferring with GLobus

Activity

Transfer ID: d3036f0c-251b-11e9-9835-0262a1f2f698
Owner: @globusid.org
Source: bilelhadri
Destination: Shaheen dm2

Transfer completed a few seconds ago

Files 1
Directories 0
Bytes Transferred 365.43 MB
Effective Speed 12.52 MB/s
Pending 0
Succeeded 2
Cancelled 0
Expired 0
Failed 0
Retrying 0
Skipped 0

Transfer Settings:
- verify file integrity after transfer
- transfer is not encrypted
- overwriting all files on destination
File Status notification (email and web-interface)

Activity

Task List

NERSC Cori to Shaheen cdI2
transferred completed a month ago

Overview

<table>
<thead>
<tr>
<th>Task ID</th>
<th>ec24a144-2a81-11e8-b7fa-0ac6873fc732</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Bilel Hadri (<a href="mailto:hadri@globusid.org">hadri@globusid.org</a>)</td>
</tr>
<tr>
<td>Source</td>
<td>NERSC Cori</td>
</tr>
<tr>
<td>Destination</td>
<td>Shaheen cdI2</td>
</tr>
<tr>
<td>Condition</td>
<td>SUCCEEDED</td>
</tr>
<tr>
<td>Requested</td>
<td>2018-03-18 10:56 am</td>
</tr>
<tr>
<td>Completed</td>
<td>2018-03-18 10:57 am</td>
</tr>
<tr>
<td>Files</td>
<td>1</td>
</tr>
<tr>
<td>Directories</td>
<td>0</td>
</tr>
<tr>
<td>Bytes Transferred</td>
<td>103.57 MB</td>
</tr>
<tr>
<td>Effective Speed</td>
<td>6.07 MB/s</td>
</tr>
<tr>
<td>Pending</td>
<td>0</td>
</tr>
<tr>
<td>Succeeded</td>
<td>2</td>
</tr>
<tr>
<td>Cancelled</td>
<td>0</td>
</tr>
<tr>
<td>Expired</td>
<td>0</td>
</tr>
<tr>
<td>Failed</td>
<td>0</td>
</tr>
<tr>
<td>Retrying</td>
<td>0</td>
</tr>
<tr>
<td>Skipped</td>
<td>0</td>
</tr>
</tbody>
</table>

Transfer Settings:
- verify file integrity after transfer
- transfer is not encrypted
- overwriting all files on destination

view debug data
Neser
• Access granted with justified needs
• Compiler available:
  – Cray, Intel and GNU
  – PrgEnv different than Shaheen
    • For serial,
      – use GNU (gcc/gfortran)
      – Use Intel (ifort/icc)
    • For parallel code
      – mpicc, mpif90 are the compiler for C and Fortran with GNU
      – mpiicc, mpiifort are the compiler for C and Fortran with Intel
      – ftn, cc with Cray compiler
• More info
  – https://www.hpc.kaust.edu.sa/neser
File Systems

- Neser is connected directly to Shaheen Lustre Parallel filesystem

Running Jobs: Two queues partitions:

- default workq (similar to Shaheen)
- tesla for GPU node

FAQs

- To launch jobs inside your job scripts, you can use srun and mpirun. Make sure you use the same mpirun launcher from the MPI library used.
- The compute nodes are exclusive, meaning that even when all the resources within a node are not utilized by a given job, another job will not have access to these resources.
- The jobs are limited to a maximum of 4 nodes with up to 2 jobs running. The remaining jobs will stay in the queue.
- To access large memory node, specify in the job script #SBATCH --mem=200G or larger
Tips and 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 (MKL)
    - Optimizing the cache and memory

Bilel Hadri – Shaheen Workshop Jan 2019
• Don’t Reinvent the wheel

• 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
Reminders

• Shaheen is a shared resource
  – Be kind to your neighbor users
  – Don’t run on login.
• Use adequately your allocation
  – Check your usage, sb kxxxx ,sb_user kxxxx
  – Prepare in advance the project proposal
• Security
  – Don’t share your account with others.
• Need help: send a ticket help@hpc.kaust.edu.sa
  – Help us to help you :D Provide details:
    • Which HP 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.
THANKS!

bilel.hadri@kaust.edu.sa
@mnoukhiya  @KAUST_HPC