# Modern HPC Tools

**PEARC 2023**
*July 2023*

**PRESENTED BY:**

***Robert McLay***

***TACC HPC***

# Please Note:

- These modern tools are portable and effective on almost all supercomputers around the world.
- Almost all tools covered in this presentation are open-source. Most of them can be installed without system administrator privileges.
- Please feel free to ask for access or installation assistance when necessary.
- Demos and labs are available to all attendees.

# Tools Make a Difference on HPC

- Great tools make a profound difference
- Require less effort to achieve some desired goals
- Save a lot of time and energy for both new and experienced users
- Enhance the user experience particularly on large-scale systems
- Under active development by experienced TACC members
- (Most of them) Available under open source licenses to public

# Outline

- User Environment
  - Lmod, SanityTool
- Workflow Assistance
  - ibrun, Launcher, Launcher-gpu, Pylauncher
- Job Monitoring
  - core_usage, show_affinity, amask,
- Runtime resource monitoring
  - Remora

# Lmod

# Manage your environment on a Supercomputer

# User Environment (1)

There are **environment variables** for defining values used by the shell (*e.g.*, bash, tcsh) and programs executed on command line.

An **environment management package** provides a command-line interface to manage the collection of environment variables associated with various software packages, and to automatically modify environment variables as needed.

# User Environment

- Environment Variables (mostly)
  - PATH (where to find command)
  - MANPATH (where to find help)
  - LD_LIBRARY_PATH (where compilers find libs, like MKL, etc.)
  - Package environment variables (TAU_METRICS, etc.)
  - Site environment variables for package (TACC_NETCDF_LIB)
- Functions and aliases
- Other possibilities: anything “unixy”

Lmod

# Lmod

- A Lua based module system
- A convenient way to dynamically change the users' environment through modulefiles.
- Add or remove environment variable easily
- Handle MODULEPATH hierarchical problem for complicated user environment
  - Only have one version active
  - Only load one compiler or MPI stack at a time

# Basic Module Commands (1)

```
# List the modules already loaded
$ module list

# Show what modules are available to be loaded
$ module avail

# Load a package
$ module load matlab

# Unload a package
$ module unload matlab
```

# Basic Module Commands (2)

```
# Change from impi to mvapich2
$ module sw impi mvapich2
```

```
# Go back to an initial set of modules
$ module reset
```

```
# Access a modulefile’s help
$ module help lammps
```

```
# Show the description section of a module
$ module whatis petsc
```

# ml: A Convenient Tool

```
# This means module list
$ ml

# Module load and unload
$ ml matlab
$ ml -matlab

# Do it in one single line
$ ml netcdf hdf5 -gsl
```

# Save/load Your Own Collection (1)

```
# Save the designed collection of modules
$ module save

# Restore the designed collection
$ module restore

# List the collections
$ module savelist
```

# Save/load Your Own Collection (2)

Users can have as many collections as they like.

```
# Save to a named collection
$ module save my_collection
```

```
# List the contents of a collection (default)
# module describe
```

```
# Restore that named collection with
$ module restore my_collection
```

# Define and Use Your Own Modulefiles

Define your own module files

- Start with an existing modulefile
- Easy to share with your colleagues

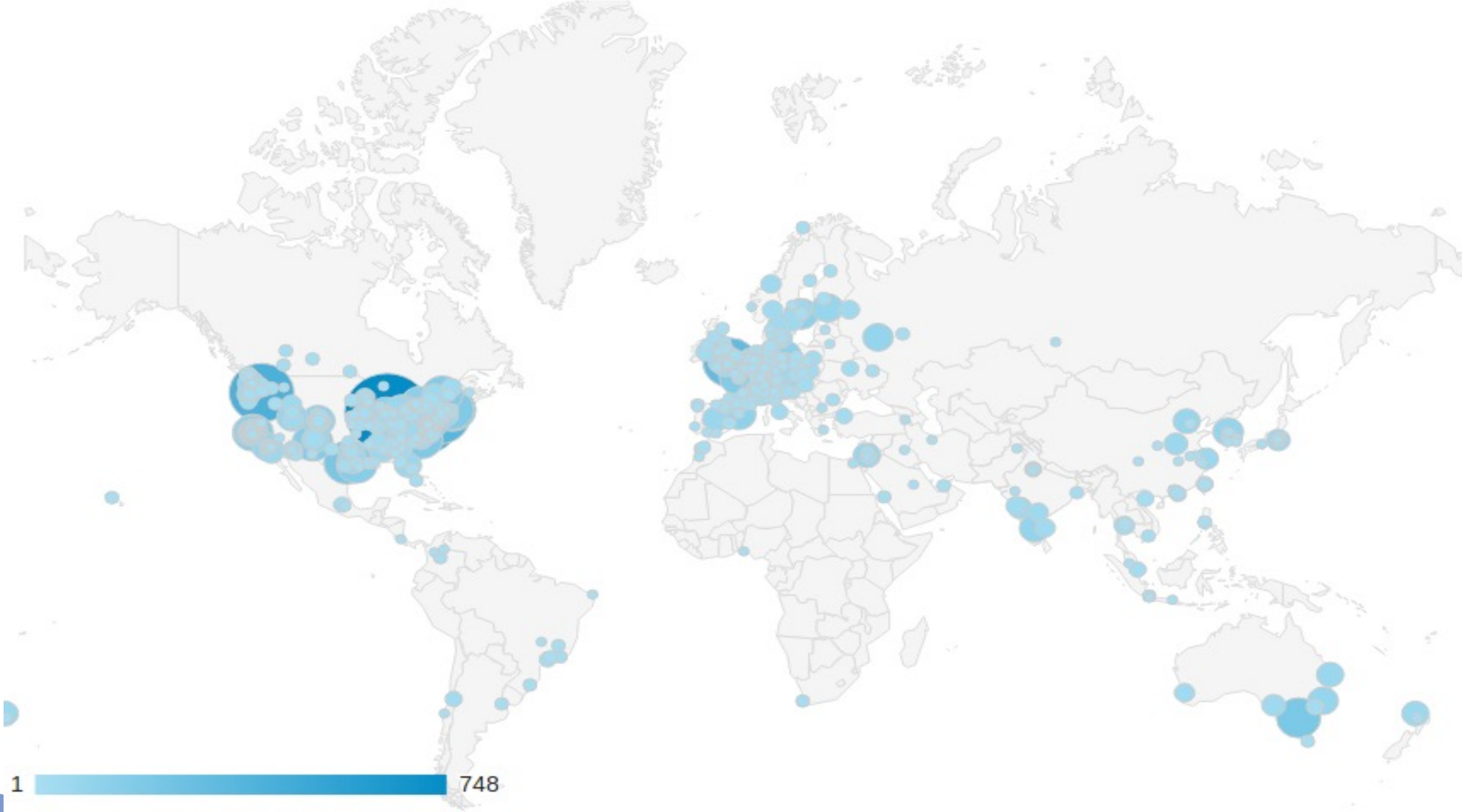
```
$ module use /scratch1/01255/siliu/mvapich2/modulefiles/
$ module load mvapich2-test/2.x-intel19
```

# Create Your Own Modulefile

- Start with existing one built by a module expert

- An introduction of writing Modulefiles:
  https://lmod.readthedocs.io/en/latest/015_writing_modules.html

- Mkmod: A tool automatically creates a modulefile
  https://github.com/milfeld/mkmod


1
748

# Lmod References

- Lmod Documentation
  https://lmod.readthedocs.io

- Monthly discussion via Zoom: See https://github.com/TACC/Lmod/wiki/home

- TACC/Lmod on github
  https://github.com/TACC/Lmod

# SanityTool
# Make my user environment valid

# Why SanityTool

- Improper or incorrect user account configurations slow down/impede work progress
- These problems could be difficult to detect (or remember), but not difficult to fix most of the time
- There are so many tools and scripts at each site, each focusing on a few tests

A lightweight integrated tool to diagnose and resolve these problems is necessary

# SanityTool

- A lightweight generic and integrated tool
- Free and open-source software
- Created in a relatively standardized format
- Contains many useful and practical tests
- Can be conveniently used whenever necessary

SanityTool

# Running SanityTool

```
$ module load sanitytool
$ sanitycheck  --help
```

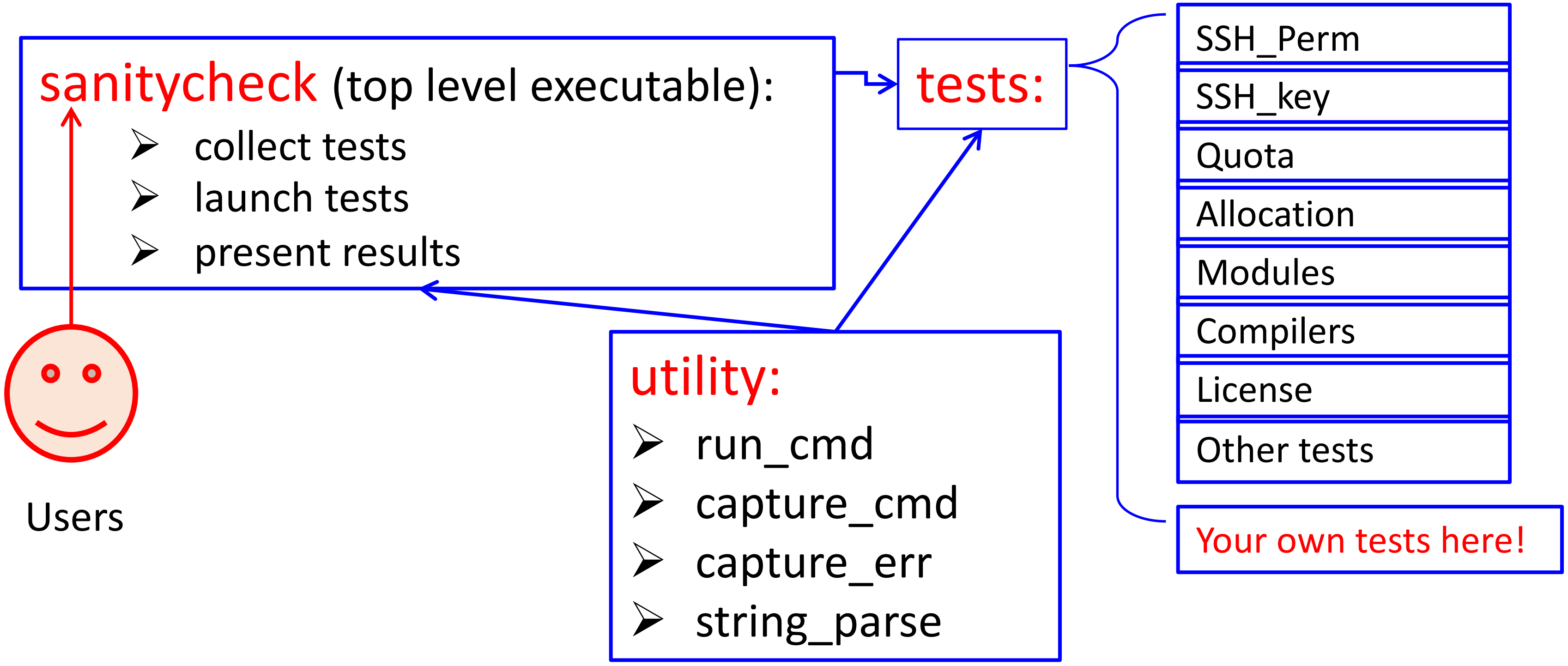
```
Sanity Tool Version:  2.0
Texas Advanced Computing Center
High Performance Computing Group
  [-h, --help]          Help information
  [-s, --silent]        Silent mode
  [-v, --verbose]       Verbose mode (default)
```

```
1: Check SSH permissions:
                  Failed
                  Error: group permission on $HOME will cause RSA to fail!
                  Error: other permission on $HOME will cause RSA to fail!
                  Make sure you have a .ssh directory under your $HOME directory.
                  You can use the following commands to set the proper permissions:
                  $ chmod 700 $HOME   #(750 and 755 are also acceptable)
                  $ chmod 700 $HOME/.ssh
                  $ chmod 600 $HOME/.ssh/authorized_keys
                  $ chmod 600 $HOME/.ssh/id_rsa
                  $ chmod 644 $HOME/.ssh/id_rsa.pub
 2: Check SSH keys:
                  Passed
 3: Check environment variables (e.g. HOME, WORK, SCRATCH) and file system access:
                  Passed
 4: Check user's queue accessibility (Stampede2 Only):
                  Passed
 5: Check allocation balance:
                  Warning: One of your projects 'ABC-123' has negative balance -1511.194.
                  Passed
 6: Check quota for $HOME and $WORK spaces:
                  Passed
 7: Check module environment:
                  Passed
 8: Check compilers:
                  Failed
                  Error: Compiler icc is not available at this time!
                  Error: Compiler icpc is not available at this time!
                  Error: Compiler ifort is not available at this time!
                  Please check your $PATH again, compilers are missing.
                  If you unload the compilers on purpose, please ignore this test.
 9: Check scheduler commands:
                  Passed
 -----------------------------------------------------
   2(out of 9) failure in sanitycheck.
 -----------------------------------------------------
```

# SanityTool Features

- Applicable to almost all supercomputer systems
  (and personal computers)
- Independent of system configurations or settings
- Work for different kinds of shell (bash, csh, zsh, etc.)
- Easy for users to remember and run
- Flexible to be run almost any time
- Full of practical tests (and still extending)

# Overall Design



**sanitycheck** (top level executable):

- collect tests
- launch tests
- present results

Users

**tests:**

- SSH_Perm
- SSH_key
- Quota
- Allocation
- Modules
- Compilers
- License
- Other tests

Your own tests here!

**utility:**

- run_cmd
- capture_cmd
- capture_err
- string_parse

# Currently Supported Tests

**Generic Tests:**

Valid ssh configurations
File system accessibility
Proper permission of file systems
Usage and quota of file systems
Necessary software licenses
Current module environments
… …

**Customized Tests:**

Necessary preloaded modules
Necessary compiler commands
Necessary scheduler commands
Whether the user is blocked
Users' allocations and balance
Permission to access to protected data
… …

# Customized Testset

The new sanitytool version 2.0 allows users to create/use their own tests.

```
$ sanitycheck –t mytestdir
```

Create  your own test case as simple as:

```
def execute(self):
    Flag=True
    output=capture(“type h5copy”)
    if "not found" in output:
        Flag = False
        self.error_message+="        ERROR: h5copy is not available!”
    return Flag
```

# Obtain SanityTool

- Obtain the source code of Sanity Tool

  https://github.com/siliu-tacc/sanitytool
- Make sure “python” and “sanitycheck” are accessible
- Go through the tests directory and choose proper tests
- Add more tests modules when necessary
- Run the “*sanitycheck*” command

# 1st hands-on/homework session: LMOD and SanityTool

# LMOD Lab (A):

- Display all available modules on the Frontera system
- View the help information for any specific module if necessary
- Load a few modules you will need for your research
- Make the new collection as the default

# LMOD Lab (B):

- Learn more about the Mvapich2 module
- Run “echo $MPICH_HOME”
- Switch to mvapich2 from impi
- Run “echo $MPICH_HOME” again

# Sanitytool Lab:

- Load the “sanitytool” module
- Run “sanitycheck” in your account
- Run “whyblockme” in your account
- Load the “sanitytool” module
- “unset SCRATCH” and run “sanitycheck” again