TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

SanityTool : *Lightweight Diagnostics for Individual User Accounts on Supercomputer Systems*

Si Liu, Robert McLay, Doug James

High Performance Computing
Texas Advanced Computing Center

PRESENTED BY:

Si Liu

HUST 16, Salt Lake City

SC16, Nov 13, 2016

Outline

- **Introduction to Sanity Tool**
- Design and implementation of Sanity Tool
- Case studies and lessons learned
- Future work and conclusion

User Account Configurations

User account configurations are essential,

- System-dependent
- User-customized
- Largely “invisible”
- Dynamic during daily usage

A variety of things (file system, allocation, quota, ssh connections) could go awry …

Why Sanity Tool

- Improper or incorrect user account configurations slow down/impede work progress
- These problems could be difficult to defect (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

Sanity Tool

- 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

Running Sanity Tool

login1> *sanitycheck --help*

Sanity Tool Version: 1.3

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 (Stampede 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.

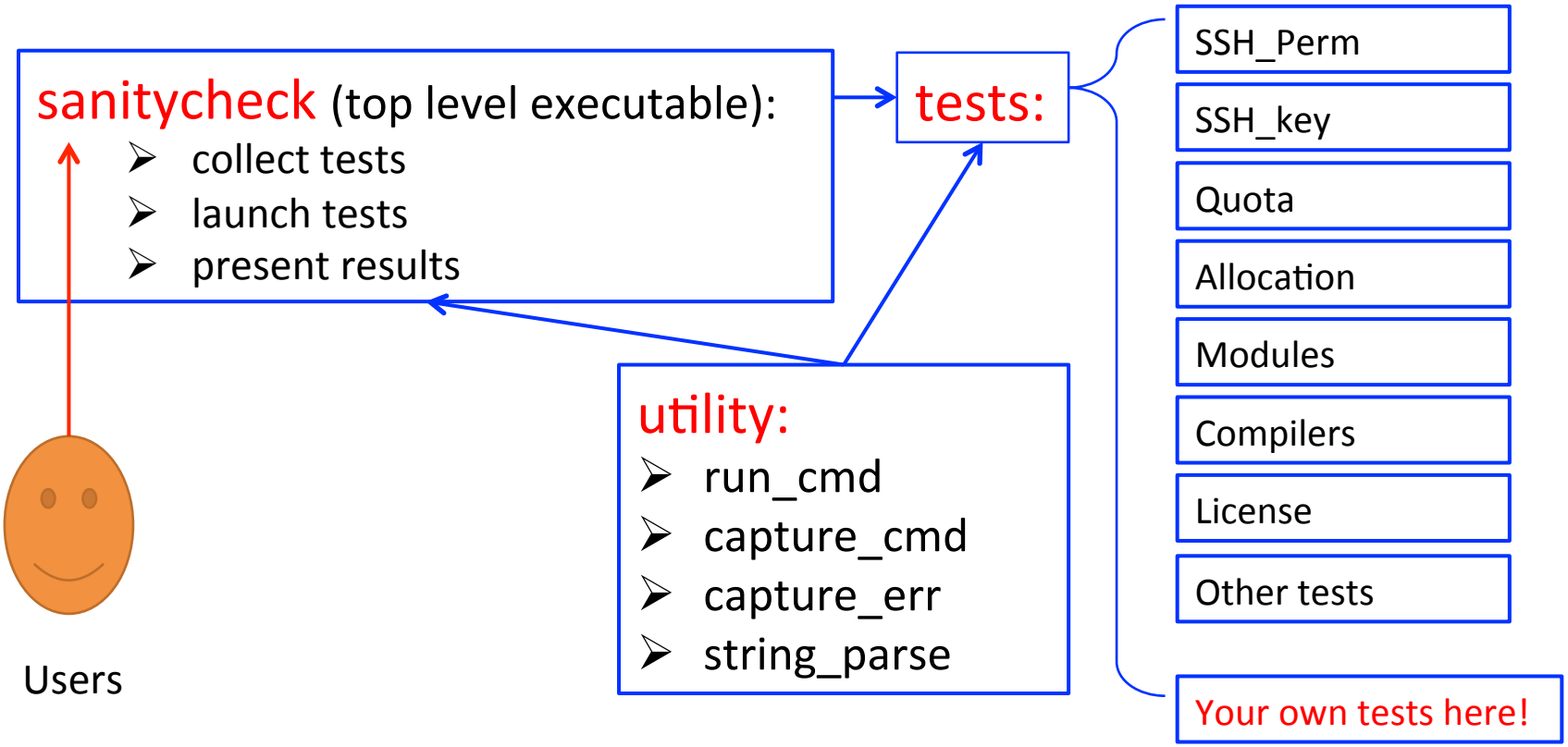
Sanity Tool Features

- 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)

Outline

- Introduction to Sanity Tool
- **Design and implementation of Sanity Tool**
- Case studies and lessons learned
- Future work and conclusion

Overall Design



Modularized Test Class

```
class test_demo(TestBase):

    __init__(self):
    setup(self):
    name(self):
    description(self):

    execute(self):
    error(self):
    help(self):

    private data when necessary
```

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

…

Obtain Sanity Tool

- 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 the tests you need
- Add more tests modules when necessary
- Run the “*satinycheck*” command

Outline

- Introduction to Sanity Tool
- Design and implementation of Sanity Tool
- **Case studies and lessons learned**
- Future work and conclusion

Case Studies

Sanity Tool is exceptionally helpful:

- Right after a user logs on a system
- Before productive jobs are submitted
- In the middle of a series of simulations
- When a new configuration is created and employed

Common MPI Failure

```
TACC: Starting up job 7292960
TACC: Setting up parallel environment for MVAPICH2+mpispawn.
TACC: Starting parallel tasks...

[c531-901.stampede.tacc.utexas.edu:mpirun_rsh][child_handler]
Error in init phase, aborting!
(0/1 mpispawn connections)
TACC: MPI job exited with code: 1

TACC: Shutdown complete. Exiting.
```

Common MPI Failure

```
TACC: Starting up job 7292960
TACC: Setting up parallel environment for MVAPICH2+mpispawn.
TACC: Starting parallel tasks...

[c531-901.stampede.tacc.utexas.edu:mpirun_rsh][child_handler]
Error in init phase, aborting!
(0/1 mpispawn connections)
TACC: MPI job exited with code: 1

TACC: Shutdown complete. Exiting.
```

This error is actually caused by a broken ssh key pair.
The design of ssh "hides" the real failure information on purpose.

Mysterious Vtune Error

TBRW_error_string returned error "Unknown reason"
TBRW_close_returned error "Unknown reason"
amplxe: Error: Unknown reason

Mysterious Vtune Error

```
TBRW_error_string returned error "Unknown reason"
TBRW_close_returned error "Unknown reason"
amplxe: Error: Unknown reason
```

This error is actually caused by a full file system.
Many programs redirect or skip the error or warning messages.

Job Management

Complicated workflow:

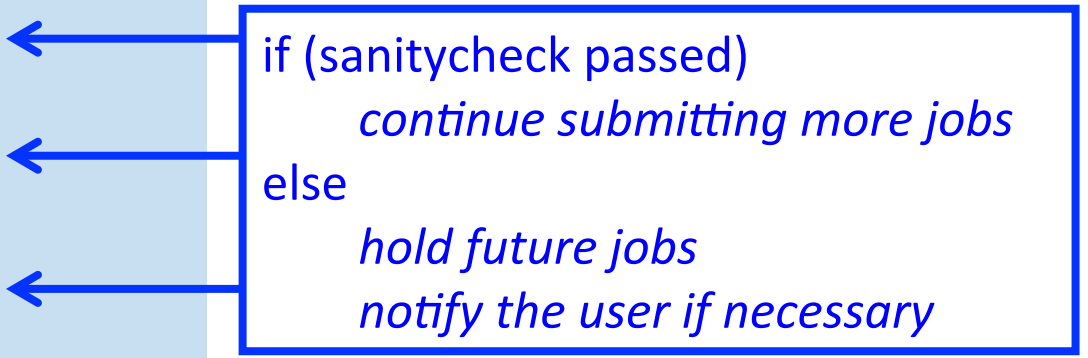
Pre-processing
… …
Submit Job A1, A2, A3 …
Submit Job B1, B2, B3 …
Submit Job C1, C2, C2 …
… …
Post-processing
Other works

This workflow is OK,
if everything is fine.
But …

Job Management

Complicated workflow:

Pre-processing
… …
Submit Job A1, A2, A3 …
Submit Job B1, B2, B3 …
Submit Job C1, C2, C2 …
… …
Post-processing
Other works



Permission Control

- More and more shared projects on supercomputers
- Applications or programs with specific license and permission

  Pre-defined target permission for binaries or data files

  Permission control for license or data manager

Manage the permission of target files through Sanity Tool:

```
perm = bool(st.st_mode & ( stat.S_IXUSR |
                stat.S_IRGRP | stat.S_IWGRP | stat.S_IXGRP |
                stat.S_IROTH | stat.S_IWOTH | stat.S_IXOTH ))
```

Outline

- Introduction to Sanity Tool
- Design and implementation of Sanity Tool
- Case studies and lessons learned
- **Future work and conclusion**

Future Work

- Add more generic and customized tests collected from our users and partners
- Integrate Sanity Tool tightly into job management programs and applications
- Embed Sanity Tool into the web-interface of user support and ticket system
- Build a semi-automatic user support system with Sanity Tool

Conclusion

- Designed for maintaining correct/reasonable user account configurations on modern supercomputers
- A set of well-designed generic and system-customized tests
- Conveniently portable to other supercomputer systems
- Easily extendable with more customized tests
- Save a tremendous amount of time and effort in diagnosing and resolving inappropriate user account configurations

Acknowledgement

- National Science Foundation:
  Stampede grant ACI-1134872
- Extreme Science and Engineering Discovery Environment:
  XSEDE grant ACI-1053575

- All members of the High Performance Computing group
  and the Advanced Computing system group
- All early users of Sanity Tool

Questions?