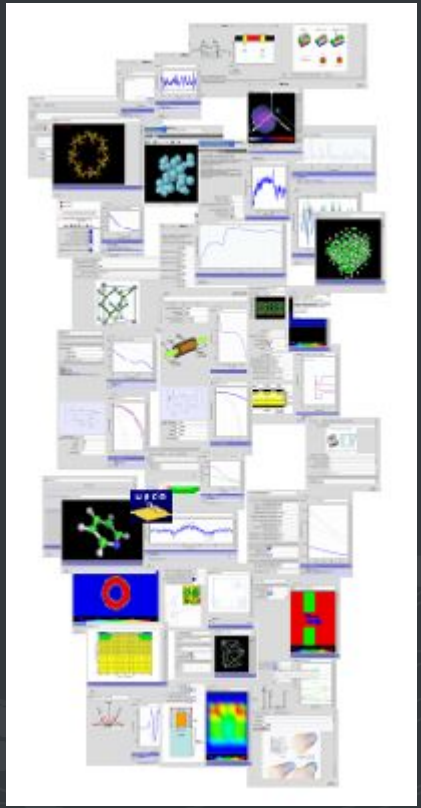
Enabling BOINC as a nanoHUB Computational Platform

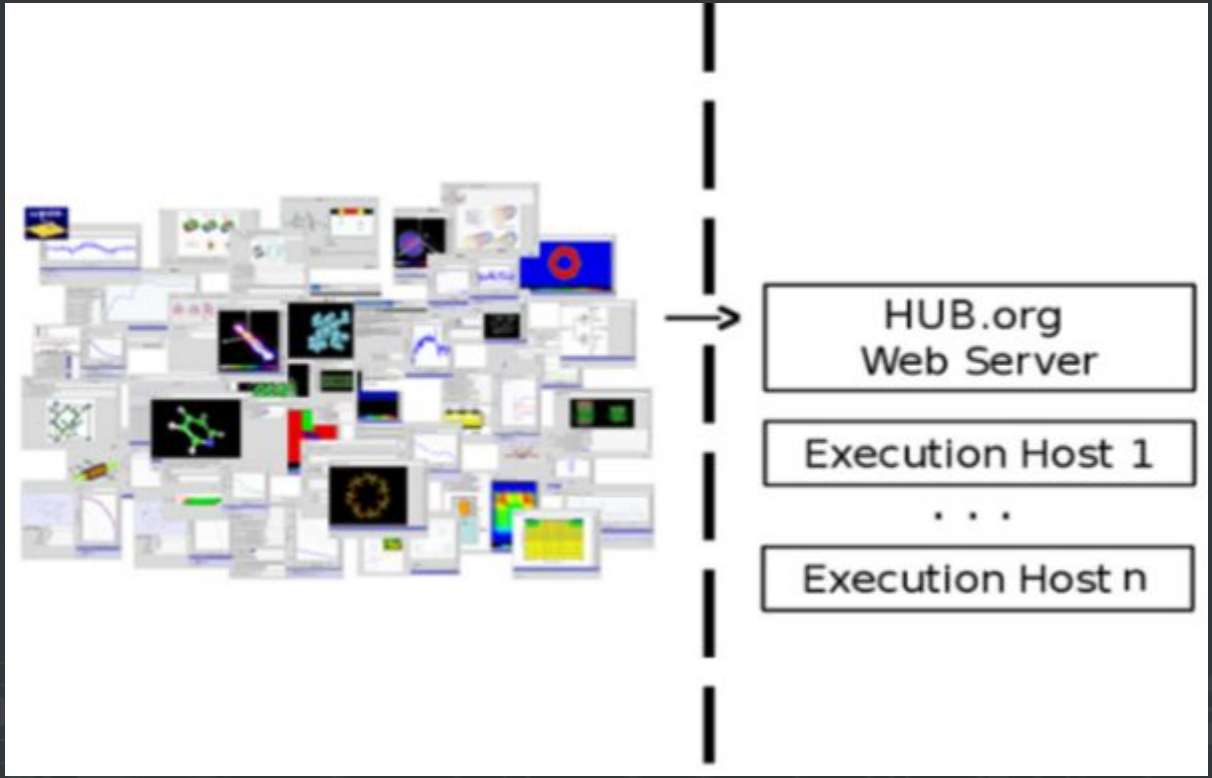
Steven Clark
nanoHUB.org
HUBzero.org
Research Computing, Purdue University

PLATFORM FOR COLLABORATIVE SCIENTIFIC COMPUTATION

- User perspective
  - Production level code
  - Powerful computing resources
  - No downloading, no compiling, ...
  - Automatically runs most updated version
  - Access regardless of location
- Developers perspective
  - GUI development environment - RAPPTURE
  - Source code management - subversion
  - Rich development platform



SUBMITTING JOBS TO LOCAL RESOURCES



SUBMIT RUN

- Execution on local host

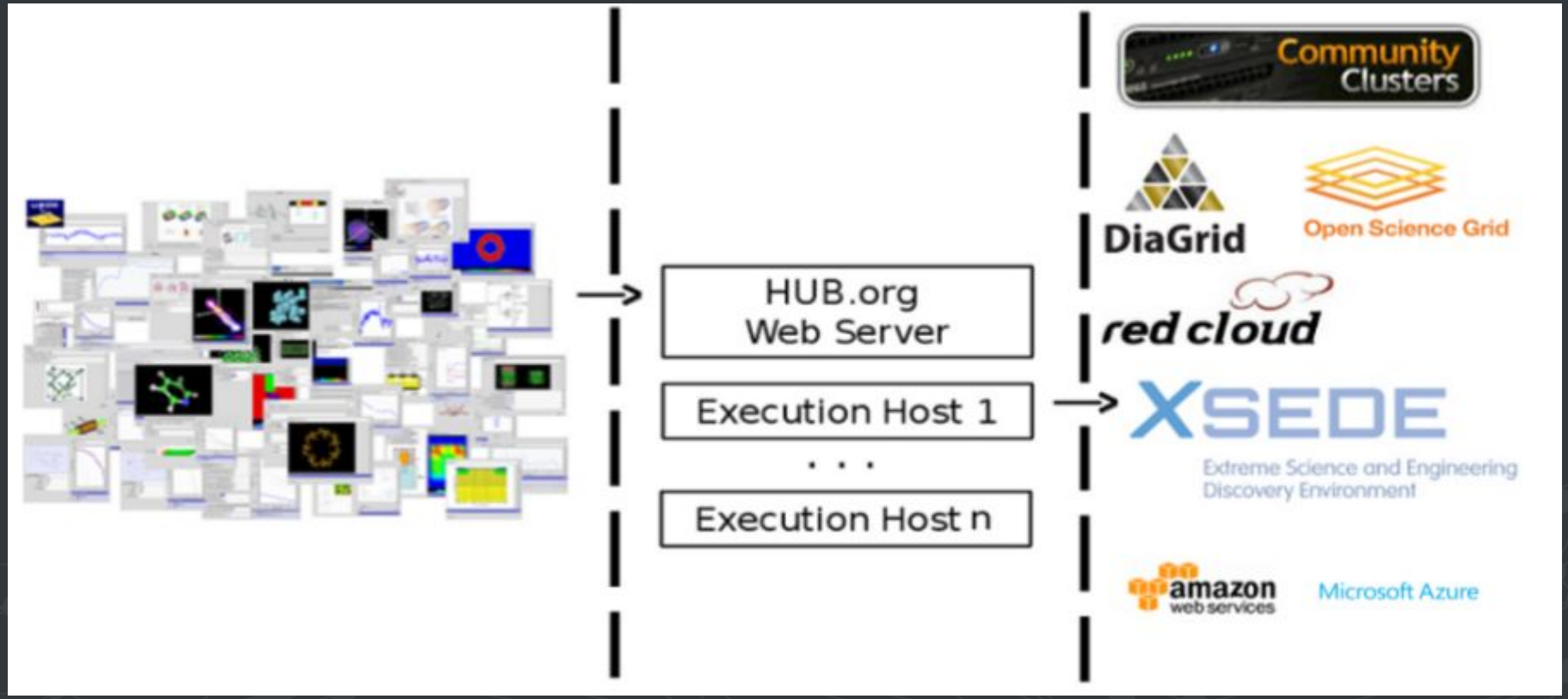
```
$ mpirun -n 16 namd2-2.9 prog.namd
```

- Execution on foreign host

```
$ submit --nCpus 16 --wallTime 10 \
         --inputfile par_all27_prot_lipid.inp \
         --inputfile ubq_ws.pdb \
         --inputfile ubq_ws.psf \
         namd2-2.9 prog.namd
```

- This command will request sixteen cores for ten minutes to run namd2-2.9 where it is installed
- Submit deduces that the file prog.namd needs to be transferred for the job to run
- Additional files that need to be transferred are specified by additional command line arguments
- Upon job completion all files created or modified by the job will be returned to the user

SUBMITTING JOBS TO FOREIGN RESOURCES



SUBMIT CONFIGURATION - OVERVIEW

- **Sites** - core set of parameters for remote resources
- **Aggregators** - mechanism for grouping multiple sites for the purpose of setting limits on job submission and prioritizing users
- **Tools** - specific set of parameters for individual tools
- **Managers** - commands to run before and after application execution
- **Identities -** configuration parameters for managing shared community credentials
- **Monitors** - parameters for configuring job tracking monitors located on remote resources
- **Appaccess** - parameters used to manage who can execute which applications on remote resources
- **Environmentwhitelist -** permissible user setable environment variables
- **Tunnels** - used to facilitate ssh tunnelling to remote resources

SUBMIT CONFIGURATION - SITES

- Core set of parameters for remote resources

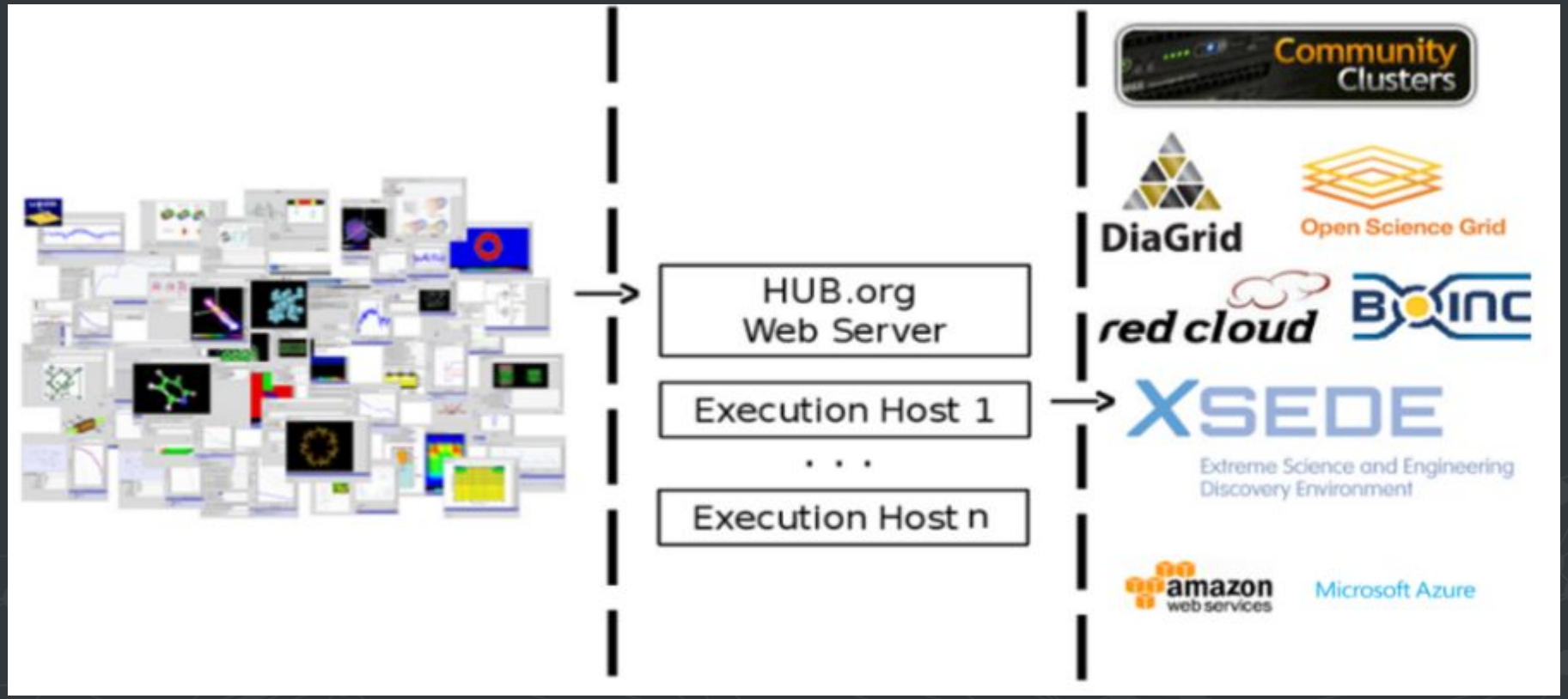
```
[normal@stampede]
venues = stampede.tacc.xsede.org
venuePort = 2222
maximumCores = 288
remotePpn = 16
remoteBatchAccount = TG-ASC140014
remoteBatchSystem = SLURM
remoteBatchPartition = normal
venueMechanism = gsissh
identityManagers = XSEDE
remoteBinDirectory = ${HOME}/Submit/bin
remoteScratchDirectory = /scratch/03280/diagrid3/diagridJobs
siteMonitorDesignator = stampede
executableClassificationsAllowed = staged, home
checkProbeResult = False
logUserRemotely = True
```

SUBMIT CONFIGURATION - TOOLS

- Specific set of parameters for individual tools

```
[namd2-2.9]
destinations = normal@stampede
executablePath = /home1/03280/diagrid3/HUBapps/share64/namd/namd-2.9/bin/namd2
remoteManager = namd-2.9
```

SUBMITTING JOBS TO FOREIGN RESOURCES



SUBMIT/BOINC - INTEGRATION

- **nanoHUB Application**
  - UI
  - User supplied data
  - Application files
- **Submit Server**
  - Common interface between local and remote resources
- **BOINC Server**
  - Job execution manager for all BOINC submissions
- **Volunteer Host**
  - Where the work happens

BOINC - APPROACH

- **Volunteer Host**
  - VirtualBox - nanoHUB applications run in Linux environment. VirtualBox provides access to Windows and MAC volunteer hosts.
  - boinc2docker - introduction of docker containers allows simpler change management. One docker container can support many nanoHUB applications.
  - Mounted volumes - allow for reduced memory requirement when loading docker container
  - nanoHUB application files sent as tar balls and are not removed at job completion to reduce bandwidth requirement
  - User supplied data is also sent as a tarball but is job specific and is removed at job completion

BOINC - APPROACH

- **BOINC Server**
  - stage_docker_image - combines docker save and stage_file to place tarballs in the download directory. Also creates nanoHUB specific vbox_* and boinc_app_* files.

BOINC - APPROACH

- **Submit Server**
  - submit_api - submit one or more jobs in a batch
  - Set of standard submit scripts for each batch system
    - receiveinput
      - createBatch
      - uploadFile
      - uploadFiles
    - submitbatchjob
      - submitBatchJob
      - submitBatchJobs
    - transmitresults
      - fetchBatchOutput
    - cleanupjob
      - retireBatch.py
    - killbatchjob
      - abortBatch.py

BOINC - APPROACH

- **nanoHUB Application**
  - Nothing new is required

SUBMIT CONFIGURATION - NEW ADDITIONS

- **Sites** - core set of parameters for remote resources
- **Aggregators** - mechanism for grouping multiple sites for the purpose of setting limits on job submission and prioritizing users
- **Tools** - specific set of parameters for individual tools
- **Managers** - commands to run before and after application execution
- **Identities -** configuration parameters for managing shared community credentials
- **Monitors** - parameters for configuring job tracking monitors located on remote resources
- **Appaccess** - parameters used to manage who can execute which applications on remote resources
- **Environmentwhitelist -** permissible user setable environment variables
- **Tunnels** - used to facilitate ssh tunnelling to remote resources
-
- **ToolFiles** - list of job independent files required for each tool
- **DockerImages** - list of docker container tar ball files required to load a container.

SUBMIT CONFIGURATION - BOINC SITE

- Core set of parameters for remote resources

```
[boinc]
venues = submit.nanohub.org
remotePpn = 1
maximumCores = 1
remoteBatchSystem = BOINC
remoteUser = USER
venueMechanism = local
remoteBinDirectory = /var/gridman/submit/bin/Boinc
executableClassificationsAllowed = staged
remoteManager = serial
siteMonitorDesignator = devboinc
checkProbeResult = False
identityManagers = user
```

SUBMIT CONFIGURATION - BOINC TOOL

- Specific set of parameters for individual tools

```
[adept_r32]
destinations = boinc
executablePath = /apps/adept/r32/middleware/invoke
toolFiles = nanohub_apps_adept_r32
remoteManager = boinc
```

SUBMIT CONFIGURATION - TOOLFILES

- Specific set of files required for individual tools

```
[nanohub_apps_adept_r32]
dockerImage = nanohub_apps_base:10
vboxFile = vbox_job_d54a9b325a3b90c9a499f5f2a19be8307f7bbcbb5c7e0326fc31a74962cfdf98.xml
boincAppFile = boinc_app_d54a9b325a3b90c9a499f5f2a19be8307f7bbcbb5c7e0326fc31a74962cfdf98
fileInfoAttributes = sticky, no_delete
fileRefAttributes = copy_file
appsFiles = apps_rappture_grid_tag_1.7.2-6645-2077.tar.gz, apps_adept_r32.tar.gz
```

SUBMIT CONFIGURATION - DOCKERIMAGES

- Specific set of docker image files required for container

```
[nanohub_apps_base:10]
imageFile = image_d54a9b325a3b90c9a499f5f2a19be8307f7bbcbb5c7e0326fc31a74962cfdf98.tar.manual.gz
layerFiles = layer_196609da1addf1aaaee3eb2b3d05ca86d279bb2ba436db2401965a5cc549f5a5.tar.manual.gz,
layer_1f60dc5c3f6839b0837c2775884edee7ab71f2afbff12fc6eebea90a2667968e.tar.manual.gz,
layer_4fb4b7f4e967f3b0d356f283aac0dcfa63ff48a410f31cc57a2272ff6d4481d8.tar.manual.gz,
layer_d5b6f70446583527a114339c9e926fef77509c2da7b16e27ede7df98e8520027.tar.manual.gz,
layer_42a604a82d8b606f1c55095213473fa4ca727d5fb6bd55efc73601e5e8db0418.tar.manual.gz
```

TOOL EXECUTION - ADEPT

- **invoke**

```
/usr/bin/invoke_app "$@" -C rappture -t adept
```

  - **-C rappture** - execute rappture with default arguments
    - Render UI using tool description file (tool.xml)
    - Except input from user
    - Execute simulation on demand
  - **-t adept** - run tool named adept

SUBMIT EXECUTION - ADEPT

```
driver=driver_adept_32.xml

cat > toolparameters.adept_r32 << EOFPARAMS
file(execute):${driver}
EOFPARAMS

submit --venue boinc \
       --inputfile ${driver} \
       --inputfile toolparameters.adept_r32 \
       --env HUB_SESSION=${SESSION} \
       --env TOOL_PARAMETERS=toolparameters.adept_r32 \
       adept_r32 -w headless
```

- **TOOL_PARAMETERS** - do not render UI
- **-w headless** - do not use window manager

SUBMIT USE CASES

- **On demand**
  - UI used to declare inputs for simulation
  - Command line
  - Single simulation or parametric sweep
- **Cache resolution**
  - Input (driver.xml) files are placed in a cache queue
  - External process pulls input from cache queue, does the simulation, saves the result
  - If cache result exists no simulation is required simply pull the existing result
  - Faster response time provides better user experience

SUBMIT USE CASES

- **Uncertainty quantification**
  - Inputs declared as distributions
  - Statistical methods used to determine input samples
  - A simulation is run for each sample
  - Result is a response surface model which can be used to approximate simulation
- **Exploratory simulation**
  - Allow for interactive selection of multidimensional input space
  - Automatically generate simulation input samples covering the space
  - Execute simulation for each sample

QUESTIONS & ANSWERS

?

SUBMIT PARAMETRIC SWEEPS

- Parametric sweeps via single command

```
submit --parameters @@cap=10pf,100pf,1uf sim.exe @:indeck
submit --parameters @@vth=0:0.2:5 --parameters @@cap=10pf,100pf,1uf sim.exe @:indeck
submit --parameters params sim.exe @:indeck
submit --data input.csv --parameters "@@doping=1e15-1e17 in 30 log" sim.exe @:infile
submit --parameters @@num=1:1000 sim.exe input@@num
submit --parameters @@file=glob:indeck* sim.exe @@file
```

- One simulation is run for each combination of parameters

SUBMIT CONFIGURATION - AGGREGATORS

- Mechanism for grouping multiple sites for the purpose of setting limits on job submission and prioritizing users

```
[normal@stampede]
destinations = normal@stampede
maximumActiveJobs = 1000
```

SUBMIT CONFIGURATION - MANAGERS

- Commands to run before and after application execution

```
[namd-2.9]
computationMode = mpi
preManagerCommands = . /opt/apps/lmod/lmod/init/sh, module load intel/15.0.2,
module load impi/5.0.2
managerCommand = ibrun -np @@{NPROCESSORS}
mpiRankVariable = PMI_ID
```

SUBMIT CONFIGURATION - IDENTITIES

- Parameters for managing shared community credentials

```
[user]
identityType = HUBuser

[commonSSH]
identityType = communitySSH
communityPrivateKeyPath = /opt/submit/etc/submit_rsa
userPrivateKeyFile = commonSSH_@@HUBUSERNAME
permanentUsers = gridman

[XSEDE]
identityType = x509
certificateDirectory = /var/gridman/Proxy/xsede-igtf
certFile = xsede_jobsubmission_cert.pem
keyFile = xsede_jobsubmission_key.pem
communityProxyFile = xsede_proxy.raw
communityRefreshInterval = 120
proxyGenerator = grid
personalizeMethod = copy
userProxyFile = xsede_@@HUBUSERID
refreshMethod = jobMonitor
refreshInterval = 60
permanentUsers = gridman
```

SUBMIT CONFIGURATION - MONITORS

- Parameters for configuring job tracking monitors located on remote resources

```
[stampede]
venue = stampede.tacc.xsede.org
venuePort = 2222
venueMechanism = gsissh
identityManager = XSEDE
remoteMonitorCommand = ${HOME}/Demo/monitors/stampede/monitorSLURM.py
```

SUBMIT CONFIGURATION - APPACCESS

- Parameters used to manage who can execute which applications on remote resources

```
[users]
whitelist = /apps/.*
priority = 0
classification = apps

[users]
blacklist = /apps/share(32|64)/debian7/padre/padre-2.4E/.*
priority = 99
classification = apps

[submit]
whitelist = ${HOME}/.*
priority = 0
classification = home
```

INSTALLATION ON FOREIGN RESOURCE

- Job management scripts
  - Import files
  - Submit job
  - Kill job
  - Export files
  - Cleanup job
- Job monitoring application
  - Python script for reporting all jobs status