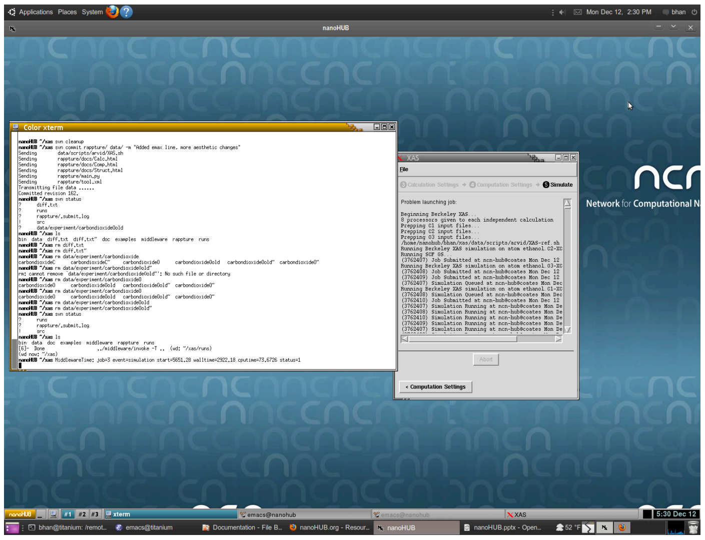
Tool development and process

1. Begin contribution process
    a. Contribute: https://nanohub.org/contribute
    b. Get workspace access (email staff for access)

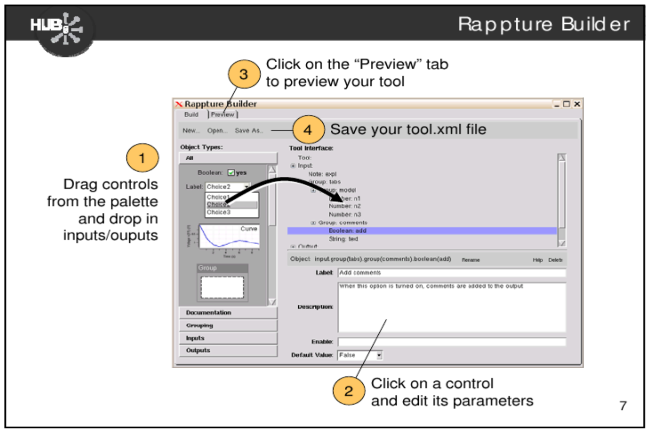


Applications Places System
nanoHUB
Color xterm
XAS
File
3 Calculation Settings → 4 Computation Settings → 5 Simulate
Abort
< Computation Settings
Network for Computational N

```
nanoHUB ~/xas svn cleanup
nanoHUB ~/xas svn commit rappture/ data/ -m "Added emax line, more aesthetic changes"
Sending        data/scripts/arvid/XAS.sh
Sending        rappture/docs/Calc.html
Sending        rappture/docs/Comp.html
Sending        rappture/docs/Struct.html
Sending        rappture/main.py
Sending        rappture/tool.xml
Transmitting file data ......
Committed revision 162.
nanoHUB ~/xas svn status
?      diff.txt
?      runs
?      rappture/.submit.log
!      src
?      data/experiment/carbondioxide0old
nanoHUB ~/xas ls
bin  data  diff.txt  diff.txt~  doc  examples  middleware  rappture  runs
nanoHUB ~/xas rm diff.txt
nanoHUB ~/xas rm diff.txt~
nanoHUB ~/xas rm data/experiment/carbondioxide
carbondioxideC      carbondioxideC~     carbondioxide0      carbondioxide0old   carbondioxide0old~  carbondioxide0~
nanoHUB ~/xas rm data/experiment/carbondioxide0old~
rm: cannot remove `data/experiment/carbondioxide0old~': No such file or directory
nanoHUB ~/xas rm data/experiment/carbondioxide0
carbondioxide0      carbondioxide0old   carbondioxide0old~  carbondioxide0~
nanoHUB ~/xas rm data/experiment/carbondioxide0
carbondioxide0      carbondioxide0old   carbondioxide0old~  carbondioxide0~
nanoHUB ~/xas rm data/experiment/carbondioxide0old
nanoHUB ~/xas rm data/experiment/carbondioxide0old~
nanoHUB ~/xas svn status
?      runs
?      rappture/.submit.log
!      src
nanoHUB ~/xas ls
bin  data  doc  examples  middleware  rappture  runs
[6]-  Done                    ../middleware/invoke -T ..  (wd: ~/xas/runs)
(wd now: ~/xas)
nanoHUB ~/xas MiddlewareTime: job=3 event=simulation start=5651.28 walltime=2922.18 cputime=73.6726 status=1
```

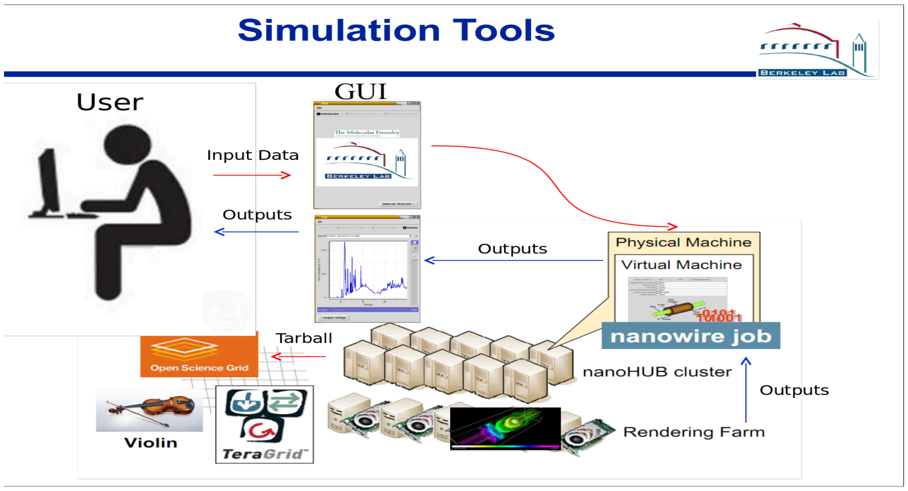
```
Problem launching job:

Beginning Berkeley XAS...
8 processors given to each independent calculation
Prepping C1 input files...
Prepping C2 input files...
Prepping O3 input files...
/home/nanohub/bhan/xas/data/scripts/arvid/XAS-ref.sh
Running Berkeley XAS simulation on atom ethanol.C2-XC
Running SCF GS...
(3762407) Job Submitted at ncn-hub@coates Mon Dec 12
Running Berkeley XAS simulation on atom ethanol.O3-XC
(3762408) Job Submitted at ncn-hub@coates Mon Dec 12
(3762409) Job Submitted at ncn-hub@coates Mon Dec 12
(3762407) Simulation Queued at ncn-hub@coates Mon Dec
Running Berkeley XAS simulation on atom ethanol.C1-XC
(3762408) Simulation Queued at ncn-hub@coates Mon Dec
(3762410) Job Submitted at ncn-hub@coates Mon Dec 12
(3762407) Simulation Running at ncn-hub@coates Mon De
(3762408) Simulation Running at ncn-hub@coates Mon De
(3762410) Simulation Running at ncn-hub@coates Mon De
(3762409) Simulation Running at ncn-hub@coates Mon De
(3762407) Simulation Running at ncn-hub@coates Mon De
```

        i. Workspace: https://nanohub.org/tools/workspace
        ii. The workspace is a virtual machine running on nanoHUB servers that can be used to test the tool in an environment that resembles that of the published tool. Developers may prefer writing code locally, but testing should be done within the workspace before updating and testing on the tool's published website. To make use of nanoHUB clusters, you must be in the workspace.
2. Subversion for source code control
    a. Using subversion: http://nanohub.org/resources/3061/supportingdocs
    b. Subversion allows several contributors to manage source code simultaneously by merging and committing local changes to a common repository version. Broken versions can also be reverted to previous versions.
    c. svn co https://nanohub.org/tools/$TOOLNAME/svn/trunk $TOOLNAME
        i. "Checkout" source to begin developing a local copy
    d. svn commit
        i. "Commit" to save local changes to repository version
3. Design GUI & script wrapper skeleton using "rappture -builder"


Rappture Builder
Click on the "Preview" tab to preview your tool
Save your tool.xml file
Drag controls from the palette and drop in inputs/ouputs
Click on a control and edit its parameters
7

a. The rappture toolkit: https://nanohub.org/infrastructure/rappture/
b. rappture builder simplifies designing and writing the GUI. drag and drop GUI elements into a tree. The builder exports the tree into "tool.xml" which rappture interprets to draw the GUI at startup. tool.xml executes the script wrapper to interpret the inputs from the user/GUI and execute code/submit jobs. Your wrapper then gathers outputs and submit them back to the GUI and user.
c. cd $TOOLDIR/rappture; rappture -builder;
   i. In python, replace variables named "str" to avoid collision with str method. Also, booleans are given values "yes" and "no", not true/false

4. Infrastructure



a. The workspace virtual machine has limited storage space and computing power and is used mainly for input/output between the user and cluster. Calculations are performed on clusters through submit *[-i $INPUTS]* $EXEC
   i. Submit: http://hubzero.org/documentation/1.0.0/tooldevs/grid.submitcmd
   ii. Berkeley XAS found best results using <4 nodes, as wait time increases significantly with the number of nodes requested
b. Submitted executables ($EXEC) must be installed by nanoHUB staff. Providing explicit instructions on compiling will expedite the process. For Berkeley XAS, a phone call made this much easier and helped develop a method for submitting

multiple executables within a single script

- i. Berkeley XAS employs a brief script (written by the main wrapper) that parses inputs, and prepares directories for the simulation. The wrapper then executes a primary script which has a call “submit -i $INPUTDAT XAS. The primary script is executed on nanoHUB clusters using $INPUTDAT (which is tarballed and sent to the cluster). The script runs several executables using the contents of the current directory and then deletes unnecessary files. Once completed, all of the remaining contents in the working directory are transferred back to the directory from which the primary script was submitted. The wrapper then parses these outputs and interprets them for the GUI and user.

c. Similarly, scripts executed by the GUI must be approved and installed by nanoHUB staff. However, scripts written on the fly by the script wrapper can be executed without prior installation.

- i. Rappture Integration with Submit: http://hubzero.org/documentation/1.0.0/tooldevs/grid.rappture_submit

d. Submit lines transfer -i $INPUTS to a new working directory on a nanoHUB cluster and begin execution. Once on the cluster, the executable will be out of contact with the user until the task is completed or fails. Finished tasks will transfer all remaining files back to the workspace. Tasks should remove unnecessary files after completion to reduce time spent transferring files (very slow otherwise).

5. svn commit; Update tool on homepage for staff installation to latest revision
6. Test within workspace using middleware/invoke

   a. Invoke: http://hubzero.org/documentation/1.0.0/tooldevs/invoke

   b. Invoke will demonstrate the behavior of the published tool, including the environment packages (assuming they are set identically). However, the path of the tool is dependent on the current directory, while the path of runs from the published tool runs from the root directory -- these may be inconsistent. Therefore relative paths should never be used (except for child directories created by the tool during an individual job).

      i. cd commands within the main script wrapper (the one called by the GUI/wrapper) do not function and do not throw errors, however cd can be used from scripts that the main wrapper executes.

   c. $TOOLDIR/middleware/invoke -T $TOOLNAME

   d. Errors in the script wrapper can also be debugged by executing the script in the shell using driver*.xml as an argument. runfiles can be generated from runs called using invoke. Running the wrapper this way will print errors and warnings that may be hidden when run using invoke.

7. Test within published webpage version

   a. Files for both published and workspace runs are accessible from the workspace in the ~/data directory. “run*.xml” files contain detailed log information regarding interactions with the GUI

8. Publish and update

Job Submission

- Use submit *[args]*
  - Submit command: http://hubzero.org/documentation/1.0.0/tooldevs/grid.submitcmd
  - *-i $INPUTFILE* to send files to cluster's working directory
- Submitted jobs are inaccessible until completed
  - jobs submitted in parallel will not communicate until task is completed and transferred back to user's directory
  - All existing files in cluster's working directory will be transfered back from cluster memory to user's current directory on nanoHUB server
    - User's have ~1.5 TB hard limit memory (~1 TB "soft" limit), exceeding hard limit will cause jobs to end abruptly
    - clean up at end of job to reduce transfer time and memory used
- Not all cluster environments are identical
  - Berkeley XAS uses Coates and Rossman exclusively

Links: Overview: http://hubzero.org/documentation/1.0.0/tooldevs
Advanced Rappture: http://hubzero.org/resources/169/download/2.2_Rappture_Advanced.pdf
Rappture FAQ: https://nanohub.org/infrastructure/rappture/wiki/FAQ
nanoHUB Tools: http://nanohub.org/resources/3865/download/tools.pdf

All of this info is relevant as of Fall 2011, however nanoHUB is frequently updated