Introduction

• Gateway = Web Graphical User Interface:
  • PIC software: BEPS, OSIRIS
  • For developers, users, educators and students

• A proof-of-concept implementation
  • From scratch
  • Ready to expand
Work Modules for PICKSC Science Gateway

- LAMP Stacks: plain PHP, no framework
- 'User' and 'Job' modules have database tables in the backend
- 'Input' interface supports for hundreds or thousands of parameters
- 'Postprocess' includes the batch processing
Current Server Architecture

- Hostname: picksc.hoffman2.idre.ucla.edu
  - *only* user: picksc
- CentOS 6.7
- PHP: 5.3.3
- MySQL 5.1
- Google sign-in enabled
- InCommon Certs
User Management: Google OAuth2

1. User’s Browser runs
2. PICKSC Web GUI Server returns request token
3. Google Authorization Server
4. PICKSC Web GUI Server returns id token
5. OAuth 2.0
6. (browser redirect)
7. User’s Browser
Input Module

• Inputs defined in a hierarchical description in XML file
  • Grouped in 3 level hierarchy
  • Can limit display subset of parameters that differ from defaults
  • Can handle multiple namelists/XMLs
  • Help available for each input

• Server implementation:
  • ‘protected’ folder
    • outside from web root
  • ‘common’
    • provide common template
  • ‘userdata’
    • user space

• Each Code should have:
  • Its own XML files
    • For multiple namelist sections
  • Its customized problems
    • Must specify/use XML filename
Smart Interface for Input

---

osiris.xml

```
<application>
  <name>osirisTest</name>
  <desc>This XML is for OSIRIS</desc>
  <command>none</command>
</application>

<input>
  <file type="text">
    <name>TEST<filename>
      <desc>The file contains namelist for OSIRIS</desc>
    </file>
    <group>
      <name>MAIN INPUT PARAMETERS</name>
      <desc>This group describes the spatial limits of the simulations</desc>
      <group>
        <name>node_conf</name>
        <param type="integer">
          <name,node_number</name>
          <desc>specifies the number of nodes to use in each direction</desc>
          <value>240,48</value>
        </param>
        <param type="boolean">
          <name,if_periodic</name>
          <value>false, true</value>
          <desc>specifies if the boundary conditions for each direction of particle species</desc>
        </param>
        <param type="integer">
          <name>grid</name>
          <value>1</value>
          <desc>spatial grid</desc>
        </param>
        <param type="integer">
          <name,time</name>
          <value>1</value>
          <desc>timestep</desc>
        </param>
        <param type="integer">
          <name,restart</name>
          <value>5</value>
          <desc>restarts</desc>
        </param>
        <param type="integer">
          <name,grid_fac</name>
          <value>8</value>
          <desc>grid factor</desc>
        </param>
        <param type="integer">
          <name,time_fac</name>
          <value>1</value>
          <desc>time factor</desc>
        </param>
      </group>
    </group>
  </group>
</input>
```
Input Validation

- All rules hard-coded in system
  - PHP + Javascript
- Chart plot:
  - Flot: Jquery-based
- In XML files:
  - `<input>
    ...
    </input>`
  - `<validation>
    <rule>3</rule>
    <rule>4</rule>
    </validation>`
Manage jobs from web server

- Insert a job record in DB to get a PK
- Create directories and batch files
- Submit the job to the scheduler
- Update the job info in DB

- Job has a separate ID (primary key) in DB
- All job info saved in `job` table in DB
- Input saved in namelist folder
- Output saved in a separate job folder
UGE Environment Setup

- 'picksc' (the apache user) is the solo user to UGE
- PHP runs bash $q$-related command
- Job submission:
  - Export bash environment variables
  - Batch cmd file:
    - Adopt IDRE script template
    - joblog & output files: in job space
  - Copy input file to job space
  - Job id: parsed from the stdout
    - Write to local DB
Job batch running to H2

- Manage files
  - Common libs in protected folder
    - Python post-process scripts
    - plot templates
  - Job-specific files: copied to job space
    - Customized parameters

- Improve the syncing process of the job status between H2 and web server.
Jobs folder

- Separate subfolder under ‘protected’ folder
- Named by the job id in DB
  - PK in jobs table
- Batch file and output files saved per job
- Deleting job will delete info in:
  1. DB
  2. Scheduler
  3. Corresponding folder
Interfaces for jobs

• General Users:
  • Run the executables from ‘common’ folder
  • Currently it can display the std out content
  • Manage their own jobs

• Admin Users:
  • Upload executables to ‘common’ folder
  • Manage all job input/outputs
Post-processing

- More options for post-processing
  - IDL, matplotlib
  - Customized parameters

- Python libs installed in /protected/libs/ folder
  - visxd from Frank for IDL
  - plot.py from Frank for matplotlib
    - osiris.plot
    - mpld3

- Online view of hdf5 pictures
A quick screenshot for reviewing the animations
Live Demo
Future Developments

- User Management
- Namelist Input Generation
- Job Management
- Post Processing

More robust method
Process more scientific problem
Submit to more remote resources
Process more data format

Migrate to professional web frameworks
New Server Architecture
Apache Airavata as a black-box middleware