Building Web Gateways to Science in Python

Shreyas Cholia
NERSC/LBL

SciPy 2010
Jun 30th 2010 Austin TX
• National Energy Research Scientific Computing Center (NERSC)
  – Supercomputing facility at Berkeley Lab in Berkeley/Oakland CA

• Mission
  – *Accelerate the pace of scientific discovery* by providing high performance computing, information, data, and communications services for *all DOE Office of Science (SC) research.*
NERSC is the Primary Computing Facility for the Office of Science

• NERSC serves a large population
  Approximately 3000 users,
  400 projects, 500 code instances
• Focus on “unique” resources
  – High end computing systems
  – High end storage systems
    • File system and tape archive
    – Interface to high speed networking
• Science-driven
  – Science problems used in machine procurements and performance metrics
  – Science services

2009 Allocations

- BES 29%
- BER 19%
- FES 18%
- HEP 17%
- ASCR 7%
- NP 10%
Diversity of Users and Systems

- Users have differing application requirements
- Wide range of access patterns
- Multiple systems to meet different user needs
Hide Complexity through Web Gateways

• Users very comfortable with web paradigm. Now expect it for usability
• Scientific Computing should be as easy online-banking
  X don’t want generic options/tools not applicable to your science
  X don’t want to deal with backend environment, UNIX CLI etc.

• NERSC gateway services
  – host the gateway
  – assist in building the webapp
  – provide building blocks to science groups for their own apps.
NERSC Science Gateways

Provides building blocks for science on the web:
- start/stop batch jobs
- manage and move data
- host data services

All through a web-browser using simple REST URLs
Python bridges the Gap

• Easy to use, expressive and productive programming language
• Strong Scientific Library Support
  – SciPy, NumPy, Scientific.IO …
• Rich web software frameworks
  – mod_wsgi + Django
• Middleware layers to access data and computation
  – pyDAP, pyGlobus
Python based Web Gateways

- **DeepSky PTF Sky Survey**
  - Image classification of Astronomical data
  - numpy for image processing

- **20th Century Re-Analysis**
  - OpenDAP interface to perform sub-selection of climate data
  - PyDAP + Scientific.IO.NetCDF

- **NEWT – NERSC Web Toolkit**
  - RESTful interface to supercomputing resources
  - Django
Goal: A gateway for selecting and manipulating telescope images (60 TB and growing)

Impact: Discovered 36 supernovae in 6 nights of data during the commissioning of the PTF Survey. The scientific gateways allowed 15 collaborators from around the world to work non-stop for the first 24 hrs during this discovery phase.
20th Century Reanalysis contains objectively-analyzed 4-dimensional weather maps and their uncertainty for most of the 1900's.

- Data stored at NERSC as NetCDF files (HDF5 format)
- PyDAP service – provides OpenDAP protocol to access subsets of data over http
- Specify URL with selection parameters – service returns dataset
- Data parsed and subselected using python Scientific.IO.NetCDF interface
Access Resources using Web API

• Encapsulate common patterns as building blocks for Science Gateways

• Building block API should be very easy to invoke eg. via a simple web page
  – Every resource should be encapsulated as a URL with a simple set of associated actions
  – Full featured web applications using Javascript + HTML5 + REST

• Science as a Service!
REST

- Representational State Transfer
- Every resource is represented by a unique http URL
- Actions are defined by standard HTTP methods: GET, POST, PUT, DELETE
- Lets you build an API that uses the language of HTTP
- NERSC Web Toolkit (NEWT) - RESTful service that provides access to NERSC resources
- NEWT combines NERSC database resources, Grid resources and other RESTful services under a single API
NEWT - NERSC Web Toolkit

- Python Django Web Service that makes HPC resources available as http URLs
- Build web applications through REST API
- No need for science team to learn underlying framework
- User interacts with a web application that exposes the necessary components of the underlying application
- Upload/download files
- Authentication
- Submit jobs to supercomputer
- Accounting information
- View Batch Queue
- Key Value Store
### NEWT API examples

<table>
<thead>
<tr>
<th>VERB</th>
<th>RESOURCE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/resource/job/</td>
<td>submit POST data to queue on R, return job id</td>
</tr>
<tr>
<td>GET</td>
<td>/resource/file/path/fname</td>
<td>get &quot;fname&quot; in &quot;path&quot; on R, copy it to apache server and download the file</td>
</tr>
<tr>
<td>GET</td>
<td>/user/username</td>
<td>get user account info</td>
</tr>
</tbody>
</table>

- Build web apps using pure HTML5/Javascript talking to NEWT service
- Mixed Backend Resources (Globus, GPFS, CouchDB, SQLLite, other Web Services) completely transparent to user
Conclusions

• The Python ecosystem allows us to create rich end-to-end interfaces to bring science to the end-user scientist over the web

• Allows us to combine Web Layer (Django, PyDAP etc.) with Scientific Computing Layer (SciPy, NumPy, PyGlobus)
Info

http://deepskyproject.org/
http://portal.nersc.gov/pydap/
http://portal.nersc.gov/newt/

Contact: Shreyas Cholia
scholia@lbl.gov