Jupyter for ATLAS experiment at BNL's SDCC

DOUG BENJAMIN
Argonne National Lab
High Energy Physics Division
Integrating Interactive Jupyter Notebooks at the BNL SDCC

D. Allan, D. Benjamin*, M. Karasawa, K. Li, O. Rind, W. Strecker-Kellogg
Brookhaven National Laboratory, *Argonne National Laboratory
BNL Scientific Data & Computing Center (SDCC)

• Located at Brookhaven National Laboratory on Long Island, NY — Largest component of the Computational Science Initiative (CSI)

• Serves an increasingly diverse, multi-disciplinary user community: RHIC Tier-0, US ATLAS Tier-1 and Tier-3, Belle-II Tier-1, Neutrino, Astro, LQCD, NSLS-II, CFN, sPHENIX….more than 2000 users from 20+ projects

• Large HTC infrastructure accessed via HTCondor (plus experiment-specific job management layers)

• Growing HPC infrastructure, currently with two production clusters accessed via Slurm

• Limited interactive resources accessed via ssh gateways
Two modes, Two workflows

- **HPC & HTC** (parallel vs interlinked, accelerator vs plain-cpu)
  - High-performance systems for GPUs / MPI / accelerators
  - High-throughput systems for big data parallel processing
- **Batch & Interactive** (working on code/GPUs vs submitting large workflows)
  - Job workflow management
  - Direct development & testing on better hardware

Traditional “Interactive SSH + Batch” paradigm places requirements on the users:
- Must be sufficiently motivated to learn and use batch systems
- Need to buy in to the workflow model: Develop, compile, move data, small-scale run on interactive nodes, full-scale processing on batch
Data Analysis As A Service

- New paradigm: **Jupyter Notebooks** (IPython)
  - Expanding the interactive toolset
  - “Literate Computing”: Combines code, text, equations within a narrative
  - Easy to document, share, and reproduce results; create tutorials...Lower barrier of entry, both for learning curve and user-base
  - Provides a flexible, standardized, platform independent interface through a web browser
  - Can run with no local software installation
  - Many language extensions (kernels) and tools available
Jupyter Service UI

Kernels

Notebook Documents

Jupyterlab
Production Architecture

• **Goal**: leverage already successful pre-existing resources, expertise, and infrastructure (batch) instead of rolling a new backend service
  ‣ Allow users to leverage any type of computational resource they might need — implies enabling both HTC and HPC/GPU, e.g. upcoming ATLAS ML workflows

• **Requirements**
  ‣ Expose to the world via unified interface [https://jupyter.sdcc.bnl.gov](https://jupyter.sdcc.bnl.gov) — common solution for HTC and HPC resource access
  ‣ Satisfy cybersecurity constraints

• **Design**
  ‣ Insert authenticating proxy as frontend to decouple jupyterhub from cybersecurity requirements (e.g. MFA)
  ‣ Scale notebooks via load-balancing as well as via batch systems
    - Automated deployment of multiple hub instances using Puppet
  ‣ Enable access to GPU nodes in a user-friendly way
    • User-specific UI for Slurm spawner support
Jupyterhub Service Architecture
Frontend Proxy Interface

- For Orchestration: a small cluster of directly-launched jupyter instances
  - HTTP-level Load-balanced from frontend proxy
  - One each on IC and HTCondor shared pool
- For Develop and Test: Use existing batch systems
  - HTCondor and Slurm support running a jupyterlab session as a batch job
  - Containers can enter at batch level to isolate external users or can be based on choice of environment
  - Best way to ensure exclusive, fair access to scarce resources (e.g. GPUs)
- Open questions: Latency, Cleanup, Starvation
Using Jupyter tools to access local resources

### Spawners

Jim Crist edited this page on Jul 16 - 26 revisions

#### Custom Spawners for JupyterHub

- **BatchSpawner** for spawning remote servers using batch systems (Torque, PBS, Slurm, etc)
- **Dockerspawner**, which actually has two different spawners in it:
  - `dockerspawner.DockerSpawner`, for spawning identical Docker containers for each user
  - `dockerspawner.SystemUserSpawner`, for spawning Docker containers with an environment and home directory for each user
- **FargateSpawner** for spawning Docker containers on AWS Fargate (ECS)
- **ImageSpawner** for allowing users to choose which Docker image to spawn.
- **KubeSpawner** for use with Kubernetes.
- **MarathonSpawner** for spawning instances on Marathon.
- **UCRSpawner** for spawning GPU instances on Marathon.
- **RemoteSpawner** (Archived, no longer maintained)
- **SimpleSpawner**, for testing purposes.
- **SystemdSpawner**, spawns notebooks with the isolation / security benefits of containers without the complexity of image management.
- **SudoSpawner** uses sudo. SystemdSpawner is recommended instead of SudoSpawner, if it is available to you.
- **WrapSpawner** allows spawner class and options to be chosen at runtime. Includes ProfilesSpawner for doing this from a spawner options form. Formerly part of BatchSpawner.
- **YarnSpawner** for spawning instances on an Apache Hadoop/YARN cluster.
Multifactor Auth

- Using Keycloak MFA tokens
- Google Authenticator or FreeOTP app
- Easy setup by scanning QR code first time
Custom Slurm Spawner Interface

* For form spawner code see https://github.com/fubarwrangler/sdcc_jupyter
Adding containers to the mix

• Use of the batch spawn allows for the use of containers
• Singularity v3.4 is used at SDCC
  • Need to convert Docker images to Singularity images
• Load the images onto local shared file system
• Custom Slurm spawner interface is extendable to pickup container location from shared file system
• Should be straight forward to use EIC containers.
Challenges of Experiment Environments

- When you get a session (start a notebook-server), which environment?
  - Customization at the kernel level or via notebook-server container
- Whose problem is setting up the environments?
  - Work for a software librarian
Orchestration: Integrating Jupyter with Compute

• How to make it easier to use compute from Jupyter?
  ‣ HTMap library from condor
  ‣ Dask / IPyParallel / Parsl etc...

• Goal: abstract away the fact that you are using a batch system at all
  ‣ Either through trivial substitutes
    - map()→htmap()
  ‣ Or through cell "magics"
    - %slurm or equivalent
  ‣ Or via nice pythonic decorators that submit to batch systems (e.g. Dask-jobqueue)

```python
from condormap import condormap
import collections
import numpy

# Sample function
def logistic(r, len=10):
    d = collections.deque(maxlen=len)
    x = 0.4
    for _ in xrange(5 * 10**7):
        x = x * r * (1.0 - x)
        d.append(x)
    return list(d)

for k, d in condormap(logistic, numpy.arange(3.5, 3.6, 0.01), withdata=True):
    print sorted(d)
    t = set(round(x, 5) for x in d)
    print k, "Mode ", len(t)
```
Conclusions

- US ATLAS worked with BNL SDCC to develop a Jupyter platform for Scientific analysis. That has grown beyond just HEP.
- The SDCC at BNL is deploying a Jupyterhub infrastructure enabling scientists from multiple disciplines to access our diverse HTC and HPC computing resources
- System designed to meet facility requirements with minimal impact on the backend
- Built-in support for experiment-based computing environment with a number of flexible access modes and workflows
- Continuing to develop new techniques for user collaboration
Additional missing enhancements for users

- Nice progress bar for a resource intensive shell would be nice to have.
- For example - CERN SWAN setup -

```
#BNL_traces.show(10)
print('number of traces - {}'.format(BNL_traces.count()))

# write out to csv file
csv_BNL_traces='csv_BNL_traces_{0:0d}_{1:02d}_{2:02d}'.format(year,month,day)
print('Writing csv output to directory - {}'.format(csv_BNL_traces))
BNL_traces.write.csv(csv_BNL_traces,mode='overwrite')

src_path = '/user/bdouglas/{}/'.format(csv_BNL_traces)
eso_dir = '/eos/atlas/nc/user/b/bdouglas/BNL_HA3/csv_traces_BNL'
dst_path = os.path.join(eso_dir,csv_BNL_traces)
if not os.path.exists(dst_path):
    os.mkdir(dst_path)

for src in os.listdir(src_path):
    cmd = 'hdfs dfs -copyToLocal {} {}'.format(src_path,dst_path)
    print(cmd)
    files = subprocess.check_output(cmd).strip()
```

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Name</th>
<th>Status</th>
<th>Stages</th>
<th>Tasks</th>
<th>Submission Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>json</td>
<td>COMPLETED</td>
<td>1/1</td>
<td>64/64</td>
<td>2 minutes ago</td>
<td>26s</td>
</tr>
<tr>
<td>1</td>
<td>count</td>
<td>COMPLETED</td>
<td>2/2</td>
<td>64/64</td>
<td>2 minutes ago</td>
<td>1m:00s</td>
</tr>
<tr>
<td>2</td>
<td>csv</td>
<td>COMPLETED</td>
<td>1/1</td>
<td>64/64</td>
<td>a few seconds ago</td>
<td>23s</td>
</tr>
</tbody>
</table>

Processing = /user/ncio01/traces/traces.2019-10-10
number of traces - 414916
Writing csv output to directory - csv_BNL_traces.2019-10-10
```
Extra Slides
This study extracts the dE/dx resolution from the sPHENIX 2019 beam test at FNAL by projecting the 16 layer prototype device to fully fledged 48-layer configuration envisioned for EIC.

Note the energy deposition from 120 GeV proton is higher than MIP due to radiative rise that leads to slightly better dE/dx resolution due to ionization statistics.

Input

```
In [1]:

const TString infile = "siso2/tga_beam_ALL-00000.ert_TrapPrototypeGasFit";
// const TString description = "Position scan 2, \eta = 0, 3, pad cut";
// const int cutout = 44 * SizeOfClusterRotated - 3;
const TString description = "Scan 2, 120 GeV proton, \eta = 0, 3, pad cut";
const TString infile = "TTrack_cluster144sizeClusterRotated-3";
// const TString infile = "eta_0.3/tga_beam_ALL-00000.ert_TrapPrototypeGasFit";
// const TString description = "120 GeV proton, \eta = 0, 3, pad cut";
// const TString infile = TTrack_cluster114sizeClusterRotated-3);
// const TString infile = TTrack_cluster144sizeClusterRotated-3);

Info in <Canvas>savebcoo++ C++ macro file: siso2/tga_beam_ALL-00000.
```

Note the cluster energy for 1-pad and 2-pad cluster (red, green) are significantly lower than that from 3+ pad clusters (blue, magenta). To be understood. For now, just analyzing 2+ pad clusters as the charge spread from zig-zag are expected to spread the charge to 3 pads.

PDF

```
In [7]:

Double_t langausf(Double_t x, Double_t *par);
{
  // Fit parameters;
  // [par[0]]: Width (sigma) parameter of Landau density
  // [par[1]]: Most Probable (MP), location parameter of Landau density
  // [par[2]]: Total area (integral = f to inf, normalization constant)
  // [par[3]]: Width (sigma) of convoluted Gaussian function
  // [par[4]]: The Landau distribution parameter (represented by the CERNLIB approximation)
  // [par[5]]: The maximum is located at x = 0.2274898 with the location parameter
  // [par[6]]: This shift is covered within this function, so that the actual
  // [par[7]]: maximum is identical to the MP parameter.

```** Notebook analysis courtesy of Jin Huang using custom sPHENIX Root Kernel**
Notebook Sharing: Short Term

- Low-effort, short-term sharing between users on the same Hub
- Sender creates shareable link that provides last saved version of notebook to link recipient
  - Short-term link expires after certain time
  - Link encodes notebook options, such as container, to ensure compatible software environment
- See https://github.com/danielballan/jupyterhub-share-link

* Courtesy Daniel Allan, illustrative gif: https://github.com/danielballan/jupyterhub-share-link/blob/master/demo.gif?raw=true
Notebook Archiving/Sharing

• Prepare a gallery of notebooks on Binder with a carefully defined software environment that anyone can recreate from a git repo with standard environment specs (e.g. requirements.txt)
  1. Enter URL of the repo
  2. Clicking "launch"
  3. Waiting and watching the build logs
  4. Copy a special link that will route directly to a Jupyter notebook running in a container that has repo contents and all software needed to run it successfully.

• Easy way for people to try your code and get running immediately

• Tightly coupled to Kubernetes and Docker, but developing similar workflows on HPC using Singularity

* Courtesy Daniel Allan
HTTP Frontend Configuration

- Authentication via Mellon plugin (for Keycloak)
- Subdivide URL space for different hub servers
  - /jupyterhub/$cluster for HTC/HPC/others
- Load-balancing configuration
  - Need cookie for sticky-sessions
  - Newest apache on RHEL7
    - Requires websockets support