Automated and Distributed Monte Carlo Generation for GlueX

Thomas Britton
CHEP 2019
What is **GlueX**?

- **GlueX** is an experiment based in **HallD**
  - One of Jefferson Lab’s four experimental halls.
  - Comprised of an international collaboration of 116 members at 27 institutions
  - Collects ~2PB per year (1 GB every second)
MC Production Line

External input

Generation

Geant

Smearing

Analysis

4-vectors

Detector response

Tuning

reconstruction

Output(s)

MCwrapper
Like all good projects, MCwrapper was born out of necessity

- Everyone had their own script(s)
- The student markets were flooded with second hand scripts with missing options
- Mirrored options across multiple configuration files leads to fraught system
- Little to no provenance for produced data

Also, I am lazy and didn’t want to have to deal with this more than once
Philosophy

- MCwrapper seeks to be a one-stop-shop for simulation for GlueX/Hall-D. It needs to be able to:
  - Complete the production chain; from generation through analysis
  - Provide basic standards of simulation
  - Allow for customization for individual studies
  - **Utilize various batch systems**
    - Often overlooked in the “trenches”
  - Provide support for new users
The Engine

• Start with core scripts to run my own workflow
  - Move to absorb colleagues’ workflows

• The engine:

  - MC.config
  - Command line
  - gluex_MC.py

  Interpret options to configure MakeMC
  Run MakeMC.(c)sh or submit to batch system

  Gcontrol.in
  And needed files
  And parameters

  Take the options and produce MC

  Output directory
Batch Systems

- New students have to learn how to interact with batch systems
  - What if MCwrapper can deal with that?
- Started locally (JLab) and branched out
  - Qsub
  - Condor
  - OSG
  - PBS
  - Slurm

*These two covered about 90% of collaborator home institutions
Includes native use of Open Science Grid.

Minimal modifications needed to run on different systems (e.g. JLab to OSG is changing a single string)
Towards Automation

• I was tired of managing “everyone’s” MC
  − Same 12 questions asked of everyone

• Stand alone, MCwrapper is extremely flexible
  − Benefit of it having no compiled code

• Any autonomous system will need to initially restrict some freedoms and cast a wide net
  − Desire an extremely low barrier to entry
    • The users should answer a minimal set of questions
With the robust engine I could wrap the wrapper
### Projects

<table>
<thead>
<tr>
<th>Progress %</th>
<th>ID</th>
<th>Email</th>
<th>Status</th>
<th>RunNumLow</th>
<th>RunNumHigh</th>
<th>NumEvents</th>
<th>Generator</th>
<th>BKG</th>
<th>RCDBQuery</th>
<th>VersionSet</th>
<th>ANA_VersionSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.00</td>
<td>589</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td></td>
<td>30381</td>
<td>30381</td>
<td>100000</td>
<td>genr8</td>
<td>None</td>
<td></td>
<td>version_recon-2017_01-ver03_8.xml</td>
<td>None</td>
</tr>
<tr>
<td>98.89</td>
<td>624</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td></td>
<td>30274</td>
<td>32000</td>
<td>10000000</td>
<td>gen_amp</td>
<td>Random:recon-2017_01-ver05</td>
<td>@is_production and @status_approved and polarization_angle==0.0</td>
<td>recon-2017_01-ver03_11.xml</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>629</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td></td>
<td>30965</td>
<td>30965</td>
<td>200000</td>
<td>genr8</td>
<td>None</td>
<td></td>
<td>version_recon-2017_01-ver03_8.xml</td>
<td>None</td>
</tr>
<tr>
<td>53.14</td>
<td>681</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td>2</td>
<td>40856</td>
<td>42577</td>
<td>20000000</td>
<td>gen_amp</td>
<td>Random:recon-2018_01-ver02</td>
<td></td>
<td>recon-2018_01-ver02_2.xml</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>688</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td></td>
<td>11366</td>
<td>11555</td>
<td>20000000</td>
<td>genr8</td>
<td>Random:recon-2017_01-ver05</td>
<td></td>
<td>recon-2017_01-ver03_11.xml</td>
<td></td>
</tr>
<tr>
<td>95.31</td>
<td>689</td>
<td><a href="mailto:user@place.com">user@place.com</a></td>
<td></td>
<td>11366</td>
<td>11555</td>
<td>20000000</td>
<td>genr8</td>
<td>Random:recon-2017_01-ver05</td>
<td></td>
<td>recon-2017_01-ver03_11.xml</td>
<td></td>
</tr>
</tbody>
</table>
Back end

• All requests are recorded in a MySQL database
  - **Projects** are then split into individual **jobs**. Each of which is recorded in the database
  - Each **job** is **attempted** perhaps many times. Each **attempt** is recorded in the database

• System’s decisions based in large part on the DB
  - Can handle managing actual running of jobs (priority system)

• Provenance achieved
  - Any simulation should have enough information saved to be recreated
Random Triggers

- Randomly triggered events taken from real data stream
  - Hit information merged with simulated events
    - More accurate picture of noise backgrounds
    - Faster to use than traditional GlueX means

- Every single attempt that requires the use of random triggers must take the data with them.
  - In HTC this could amount to TB needing to be transferred

- Use XRootD!
  - No need to take the whole file (1 GB) when the attempt may only need a few MB
Other Automations

- The system can submit automatically to multiple places
  - JLab and OSG currently

- It monitors the OSG submit node and decides on where to run jobs. The user is completely oblivious

- Provides opportunities to optimize payloads dynamically
  - If a site prefers long jobs we could scale the payload to maximize usage. We can do this dynamically and without intervention
Other Automations (cont.)

• The system knows some common problems with jobs and will correct jobs on the fly.
  − It will also automatically resubmit failed jobs if it expects them to succeed
  • There is a limit

• Not fool-proof, but lowers user effort in tracking/monitoring jobs
  − Most users fill out the form, get a receipt email, wait a short time, and receive a completion email (generally within two days even for decently sized samples)
Adoption of MCwrapper-bot

- Month over month growth!
- To date:
  - More than 52TB produced
  - 44 unique users
  - ~104k core days used
  - ~1M jobs submitted
  - 1 Billion events in ~1 week from a centralized system

I use it to run everyone’s MC
Future

• Measure basic properties of jobs during testing
  ∵ Scale requests to better obtain resources

• More sophisticated decision making
  ∵ Change how jobs are broken apart to optimize for target platforms

• Adapt to other experiments
  ∵ At its core MCwrapper(-bot) is a means of running a script with options anywhere
Conclusion

- MCwrapper(-bot) successfully runs and manages MC production at scale for GlueX
  - Front-end provides near zero barriers for entry
    - The new graduate student or collaborator can begin simulating GlueX events in ~10 minutes with built in examples

- With strong avenues for development, the system can increase production efficiency by dynamically tailoring to specific workflows and ultimately expand beyond the management of GlueX MC