Fitting and Statistical Inference as a Service

Matthew Feickert

University of Illinois at Urbana-Champaign

Mini-Workshop on Portable Inference
December 4th, 2020
Collaborators

Lukas Heinrich, CERN
Giordon Stark, UCSC SCIPP
Alexander Held, NYU

Ben Galewsky, NCSA
Ricardo Rocha, CERN
Sinclert Pérez, NYU
Accelerating fitting (reducing time to insight (statistical inference)!)  
Analysis Systems pipeline already has beta infrastructure for the final statistical inference stages with pyhf + cabinetry  
  - Ask if you have questions on these projects
Fitting as a Service with pyhf

▶ Want to leverage pyhf hardware accelerated backends at HPC sites for real analysis speedup
  ▶ Fitting time from hours to minutes
▶ HTC not target, so deploy (fitting) Function as a Service (FaaS)
  ▶ Use API to deploy fits and return JSON output

```
$ cat benchmarks/gpu/gpu_pytorch.txt
# time pyhf cls --backend pytorch HVTWZ_3500.json
{
  "CLs_exp": [
    0.07676925199218336,
    0.17262542362618583,
    0.3572332455085822,
    0.6318728762727417,
    0.879799718293609
  ],
  "CLs_obs": 0.25670190274923205
}
```

ATLAS workspace that takes over an hour on ROOT fit in under 2 minutes with pyhf on GPU
Open fields of exploration

- Early days in exploring solutions to implementation of Fitting as a Service
- Parallel explorations of what service and user API would look like
- **funcX** from Globus Labs
  - High-performance FaaS platform
  - Allows users to register and then execute Python functions in “serverless supercomputing” workflow
- **Knative**
  - Well adopted as a Serverless/FaaS solution on Kubernetes
  - Deployment model promotes efficient resource usage and simplifies bursting
### Infrastructure Perspective

**Possible workflow for development (here for funcX) and end user experience**

<table>
<thead>
<tr>
<th>Development</th>
<th>Building</th>
<th>Deploying</th>
<th>Governance</th>
<th>End users</th>
<th>Fit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pyhf</strong> evolves over time. Code on GitHub released to PyPI and conda-forge.</td>
<td><strong>FuncX</strong> encapsulation of Python functions. Images are published to a cloud registry (DockerHub?), so they can be accessed.</td>
<td><strong>Kubernetes</strong> is used to deploy the functions. High scalability plays nicely with computational expensive workflows.</td>
<td><strong>Governance</strong> model required. Someone needs to coordinate new deployments across the stack. In addition to enable / disable access through an auth DB.</td>
<td>Ask for access to the service. Given the amount of computing power the service could use, auth is required. Some ticketing procedure must be defined (GitHub issues?).</td>
<td>Users send HTTP requests. Users query the service, with some basic auth information. Service validates user auth before proceeding forward.</td>
</tr>
</tbody>
</table>

![Diagram](image-url)
funcX

- Prototype workflow for fitting models from pyhf pallet for published ATLAS SUSY 1Lbb analysis
  - Currently deployed on Chicago River HPC cluster
  - Example implementation of deployment model
- Uses Python driver with globus for authentication
- Have tested and are able to fit all models in analysis (125 signal patches) in just under 2 minutes 30 seconds
  - N.B. Wall time includes downloading pyhf pallet from HEPData, starting funcX, sending data to funcX, and fits
  - Currently CPU, but parallelization gives significant speedup
- For working prototype, this is already a win!
- Investigating workflows for pseudoexperiment generation that benefit from hardware acceleration

```bash
$ time python demo_fit.py prepare waiting-for-nodes
> <pyhf.workspace.Workspace object at 0x7efbd9955380>
> Task C1N2_wh_hbb_1000_0 complete, there are 0 results now
> Task C1N2_wh_hbb_1000_100 complete, there are 1 results now
> Task C1N2_wh_hbb_1000_150 complete, there are 2 results now
> Task C1N2_wh_hbb_1000_200 complete, there are 3 results now
> Task C1N2_wh_hbb_1000_250 complete, there are 4 results now
> Task C1N2_wh_hbb_1000_300 complete, there are 5 results now
> Task C1N2_wh_hbb_1000_350 complete, there are 6 results now
> Task C1N2_wh_hbb_1000_400 complete, there are 7 results now
> Task C1N2_wh_hbb_1000_50 complete, there are 8 results now
> Task C1N2_wh_hbb_150_0 complete, there are 9 results now
> Task C1N2_wh_hbb_150_100 complete, there are 10 results now
> Task C1N2_wh_hbb_150_150 complete, there are 11 results now
> Task C1N2_wh_hbb_150_200 complete, there are 12 results now
> Task C1N2_wh_hbb_150_25 complete, there are 13 results now
> Task C1N2_wh_hbb_150_50 complete, there are 14 results now
> Task C1N2_wh_hbb_200_0 complete, there are 15 results now
> ... skipping forward for space...
> ... Task C1N2_wh_hbb_800_50 complete, there are 114 results now
> Task C1N2_wh_hbb_900_0 complete, there are 115 results now
> Task C1N2_wh_hbb_900_100 complete, there are 116 results now
> Task C1N2_wh_hbb_900_150 complete, there are 117 results now
> Task C1N2_wh_hbb_900_200 complete, there are 118 results now
> running
> Task C1N2_wh_hbb_900_300 complete, there are 119 results now
> Task C1N2_wh_hbb_900_350 complete, there are 120 results now
> Task C1N2_wh_hbb_900_400 complete, there are 121 results now
> Task C1N2_wh_hbb_900_50 complete, there are 122 results now
> Task C1N2_wh_hbb_900_50 complete, there are 123 results now
> Task C1N2_wh_hbb_900_25 complete, there are 124 results now
> ...........................................
> ... skipping print of results
> real 2m27.249s
> user 0m12.273s
> sys 0m2.052s
```
Knative + GPU Workloads

- CERN colleagues built prototype scaling out from CERN to Google Cloud Platform (GCP)
  - Especially interesting for GPUs/TPUs
- Supports fast auto scaling of workloads (secs) and clusters (mins) to meet demand
- Working on version that allows per second reporting instead of per script execution

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: autoscale-go
  namespace: default
spec:
  template:
    metadata:
      annotations:
        # Target 10 in-flight-requests per pod.
        autoscaling.knative.dev/target: "10"
  spec:
    containers:
    - image: rochaporto/fitting:cuda11.0
```

*Example toy fit run on all available GCP cards*
Knative + GPU Workloads

- Comparison on all GCP and Azure cards for both single and double precision
- Performance comparable across multiple
t- Single precision could be used strategically

**Single precision**

<table>
<thead>
<tr>
<th>GPU Cards</th>
<th>Approximate fit time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M60 P4 P40 T4 K80 P100 V100 A100 TPU2-8</td>
<td>0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0</td>
</tr>
</tbody>
</table>

**Double precision**

<table>
<thead>
<tr>
<th>GPU Cards</th>
<th>Approximate fit time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M60 P4 P40 T4 K80 P100 V100 A100</td>
<td>0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0</td>
</tr>
</tbody>
</table>
Fitting as a Service with pyhf is a natural progression of the final stage of Analysis Systems pipeline deployed to HPC sites.

- Consumes pyhf Python API
- Allows for parallelization of fitting models from pyhf pallets across HPC/GPU systems
- Possible interface with cabinetry?

Very early stage of development, but given the relative stability of pyhf API, it should be able to develop and explore the idea space quickly.

- pyhf not yet at v1.0, but relevant API is rather stable

Working deployment on funcX, trial deployments at CERN with Knative.

End user API currently under design iteration:

- Small library to give service agnostic CLI API as well as Python API?
- Scope will dictate more as projects evolve.