

PanDA analysis queue: GOOGLE100

- Autoscaled, preemptible cluster: 2 to 10 n2-standard-8 nodes (8 core, 32 GB mem, 375GB local SSD)
- Nikolai already ran few small test tasks
- Currently empty
 - Would be good to have a rough schedule for testing and ramp up to optimize budget
 - Note: I never understood GCE autoscaling and why it doesn't scale in
- Others can use the queue as well
 - You need to ask Rucio to enable your account and provide a quota in order to use the associated Rucio Storage Element GOOGLE_EU

Dask Gateway + JupyterHub

- GKE cluster with 3 e2-standard-8 nodes (8 core, 32 GB mem, 100 GB disk)
- Rough installation available
- JupyterHub
 - Password authentication (one password for all users)
 - Theoretically it's possible to setup OAuth using [gcp4hep.org](https://github.com/gcp4hep/gcp4hep.org)
 - Jupyter accounts currently have 10 GB disk
- Dask Gateway
 - Users can create their own Dask Cluster through python and JupyterHub
 - Possible to customize worker corecount, memory, image
 - Image needs to fulfill few conditions (e.g. have dask-gateway installed)
- Nikolai and Lukas have access to start providing feedback
- I can provide others access
 - Potentially unstable at times if we decide to add any functionalities

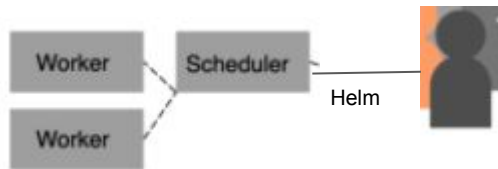
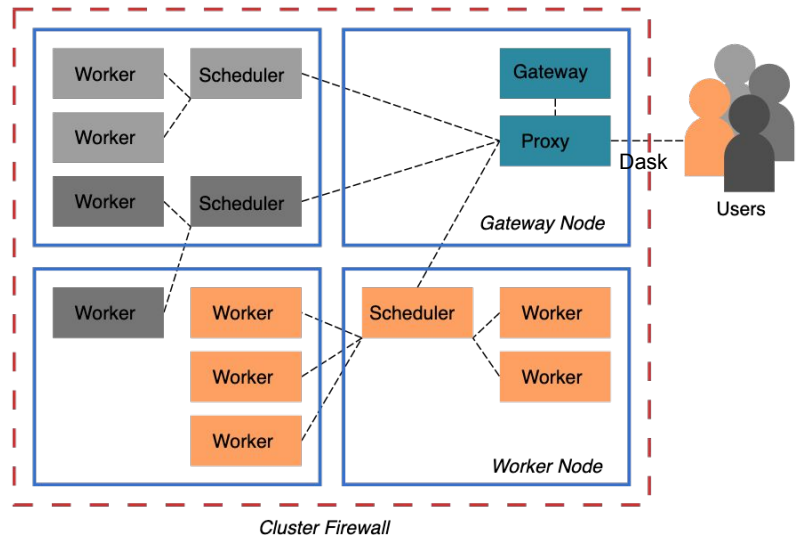
Suggestions and questions

- Other capabilities can potentially be added based on feedback

Backup

Dask-Kubernetes

- Dask Gateway for multi user cluster
 - User only has access to Dask
 - [Daskhub helm](#): (rest of slides refer to this)
 - Bundles Dask gateway + Jupyterhub
 - Available since August
 - I think this will be the favoured chart
 - [Dask gateway helm](#):
 - Independent installation of Dask Gateway
- [Dask single user](#)
 - User interacts with Kubernetes cluster and creates his own Dask clusters



JupyterHub users and basics

- Password authentication
 - Can add different users + JupyterHub admins through helm config file
 - All share the same password
 - OAuth integration using gcp4hep.org could be possible, but not first priority
- Each user runs in a separate pod
 - 10GB disk assigned
 - If it fills up, JH account stops working
 - Can be solved by resizing disk
- Running as http, not https

http://<JH IP>

GW=GateWay;
JH=JupyterHub

A web form for signing into JupyterHub. It has an orange header with 'Sign in'. Below it is a yellow warning box: 'Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.' There are input fields for 'Username:' and 'Password:', and an orange 'Sign in' button at the bottom.

http://<JH IP>/hub/admin

The JupyterHub Admin interface shows a table of users. At the top, there are tabs for 'User' and 'Admin', and buttons for 'Add Users', 'Start All', 'Stop All', and 'Shutdown Hub'. The table lists users: fbarreir (admin), dask, user1, and user2. Each user row has buttons for 'start server', 'access server', 'edit user', and 'delete user'.

User	Admin	Last Activity	Running (1)
fbarreir	admin	19 minutes ago	start server access server edit user
dask		3 days ago	start server edit user delete user
user1		2 days ago	start server edit user delete user
user2		Never	start server edit user delete user

Displaying users 1 - 4 of 4

```
[root@aipanda185 ~]# kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE
api-dhub-dask-gateway-cb84dd5d5-85mdc	1/1	Running	0	35h
continuous-image-puller-Zz4fq	1/1	Running	0	2d15h
continuous-image-puller-fm2mh	1/1	Running	0	2d15h
continuous-image-puller-smj5f	1/1	Running	0	2d15h
controller-dhub-dask-gateway-66f57c96ff-nqxzg	1/1	Running	0	2d15h
hub-8f8c65b9f-fpncv	1/1	Running	0	35h
jupyter-fbarreir	1/1	Running	0	22h
proxy-d9699f6b-pldzm	1/1	Running	0	35h
traefik-dhub-dask-gateway-cc4bd9759-qchkk	1/1	Running	0	2d15h
user-scheduler-76977c766-gvwvw	1/1	Running	0	2d15h
user-scheduler-76977c766-pcndn	1/1	Running	0	2d15h

Local clusters

The screenshot displays the Dask dashboard and JupyterLab interface. On the left, the Dask dashboard sidebar shows various monitoring tools like 'MEMORY BY KEY', 'BANDWIDTH WORKERS', 'PROFILE SERVER', 'GRAPH', 'WORKERS', 'NPROCESSING', 'GPU UTILIZATION', 'NBYTES', 'GPU MEMORY', 'TASK STREAM', 'BANDWIDTH TYPES', 'PROFILE', 'COMPUTE TIME PER KEY', 'PROGRESS', 'AGGREGATE TIME PER ACTION', 'CPU', and 'CLUSTER MAP'. The 'CLUSTER MAP' section shows a 'LocalCluster 1' with details: Scheduler Address: tcp://127.0.0.1:34963, Dashboard URL: http://127.0.0.1:8787/status, Number of Cores: 8, Memory: 33.68 GB, Number of Workers: 4. Below this are buttons for '<> SCALE' and 'SHUTDOWN'.

The main JupyterLab window shows a Python 3 notebook with the following code and output:

```
[3]: a = da.random.normal(size=(1000, 1000), chunks=(50, 50))
```

	Array	Chunk
Bytes	8.00 MB	20.00 kB
Shape	(1000, 1000)	(50, 50)
Count	400 Tasks	400 Chunks
Type	float64	numpy.ndarray

[4]: `a.sum()`

	Array	Chunk
Bytes	8 B	8 B
Shape	()	()
Count	939 Tasks	1 Chunks
Type	float64	numpy.ndarray

[5]: `a.compute()`

```
[5]: array([[ -0.84876043,  0.61908852, -1.89251837, ..., -1.49685442,
           0.14731227,  1.38812212],
 [ 1.83534814, -0.88381477, -0.79296392, ..., -0.18761566,
  1.594343 ,  1.85689245],
 [ 1.29439812,  0.06835067, -0.3935912 , ...,  0.60144869,
 -0.52183523,  0.26204039],
 ...,
 [-0.78835079, -0.44857443, -0.49627805, ..., -0.73206019,
  0.35906718, -1.73728369],
 [-0.29689929, -0.15352357, -2.30212712, ...,  0.35782569,
  0.14893276, -1.62190502],
 [-0.73430639, -0.03470112, -0.09381767, ...,  0.01730703,
  0.71722406, -0.68804543]])
```

Below the notebook, there are two panels: 'Dask Task Stream' and 'Dask Cpu'. The 'Dask Task Stream' panel shows a task stream visualization with a timeline from 0ms to 300ms. The 'Dask Cpu' panel shows a CPU utilization bar chart.

- Creates a LOCAL cluster, i.e. all dask components running inside your user's Jupyter pod
- I think there are some configuration possibilities for additional packages, but I don't think we care about LOCAL clusters, so I won't spend any time looking at configuration options

Distributed clusters

```
[1]: from dask_gateway import GatewayCluster
cluster = GatewayCluster()
client = cluster.get_client()
cluster
```

GatewayCluster

Workers 2

Cores 2

Memory 4.29 GB

▼ Manual Scaling

Workers

Scale

► Adaptive Scaling

Name: default.272757462892425096bbfeafa321c41c

Dashboard: /services/dask-gateway/clusters/default.272757462892425096bbfeafa321c41c/status

Copy this link into the Dask lab extension to enable fancy monitoring

```
[2]: import dask.array as da
a = da.random.normal(size=(1000, 1000), chunks=(500, 500))
a.mean().compute()
```

```
[2]: -0.00012254271380655166
```

- No button to create cluster(AFAIK), need to do it through python - preconfigured

```
[root@aipanda185 ~]# kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE
api-dhub-dask-gateway-cb84dd5d5-85mdc	1/1	Running	0	35h
continuous-image-puller-2z4fq	1/1	Running	0	2d15h
continuous-image-puller-fm2mh	1/1	Running	0	2d15h
continuous-image-puller-smj5f	1/1	Running	0	2d15h
controller-dhub-dask-gateway-66f57c96ff-nqxzg	1/1	Running	0	2d15h
dask-scheduler-272757462892425096bbfeafa321c41c	1/1	Running	0	44s
dask-worker-272757462892425096bbfeafa321c41c-6vrdr	1/1	Running	0	12s
dask-worker-272757462892425096bbfeafa321c41c-tzpwk	1/1	Running	0	12s
hub-8f8c65b9f-fpncv	1/1	Running	0	35h
jupyter-fbarreir	1/1	Running	0	23h
proxy-d9699f6b-pldzm	1/1	Running	0	35h
traefik-dhub-dask-gateway-cc4bd9759-qchkk	1/1	Running	0	2d15h
user-scheduler-76977c766-gvwvw	1/1	Running	0	2d15h
user-scheduler-76977c766-pcndn	1/1	Running	0	2d15h

Direct python interaction with GW

http://<JH IP>/hub/token

Request new API token

Note

Dask Gateway

This note will help you keep track of what your tokens are for.

↓

```
[root@aipanda185 gke-dask]# export JUPYTERHUB_API_TOKEN=
[root@aipanda185 gke-dask]# python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from dask_gateway import Gateway
>>> AR
>>> gateway = Gateway("http://<GW IP>/services/dask-gateway", auth="jupyterhub")
>>> cluster = gateway.new_cluster()
>>> cluster.scale(2)
>>> client = cluster.get_client()
/usr/local/lib/python3.6/site-packages/distributed/client.py:1129: VersionMismatchWarning: Mismatched versions found

-----+-----+-----+-----+
| Package | client      | scheduler  | workers    |
+-----+-----+-----+-----+
| msgpack | 1.0.2       | 1.0.0      | 1.0.0      |
| numpy   | 1.19.5     | 1.19.2     | 1.19.2     |
| python  | 3.6.8.final.0 | 3.8.3.final.0 | 3.8.3.final.0 |
+-----+-----+-----+-----+
Notes:
- msgpack: Variation is ok, as long as everything is above 0.6
  warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"]))
>>> import dask.array as da
>>> a = da.random.normal(size=(1000, 1000), chunks=(500, 500))
>>> a.mean().compute()
0.000992059770998904
```

To customize workers in your cluster:

```
gateway.new_cluster(image='fbarreir/daskgateway
:latest', worker_cores=2, worker_memory=4)
```