Deploying Infrastructure for Dask on Google Cloud

Sang-Yun Oh

Dept. of Statistics and Applied Probability

University of California Santa Barbara

What is Dask?

Dask breaks up large computations and route parts of them efficiently onto distributed hardware. Dask is routinely run on thousand-machine clusters ...

About Dask

- Description of Dask
- Example distributed applications
- Distributed infrastructure

Computing Infrastructure for Dask

- Dask run on laptops, clusters, and HPC environments
- Develop/test in small scale (laptops)
- Deploy/scale in large scale (clusters and HPC environments)

Which Infrastructure to Use?

- **Cloud computing**: slow(er) but easy access and on-demand
- **HPC**: highly optimized but access is restricted and may not be interactive

Cloud Computing and Reproducibility

What is reproducibility?

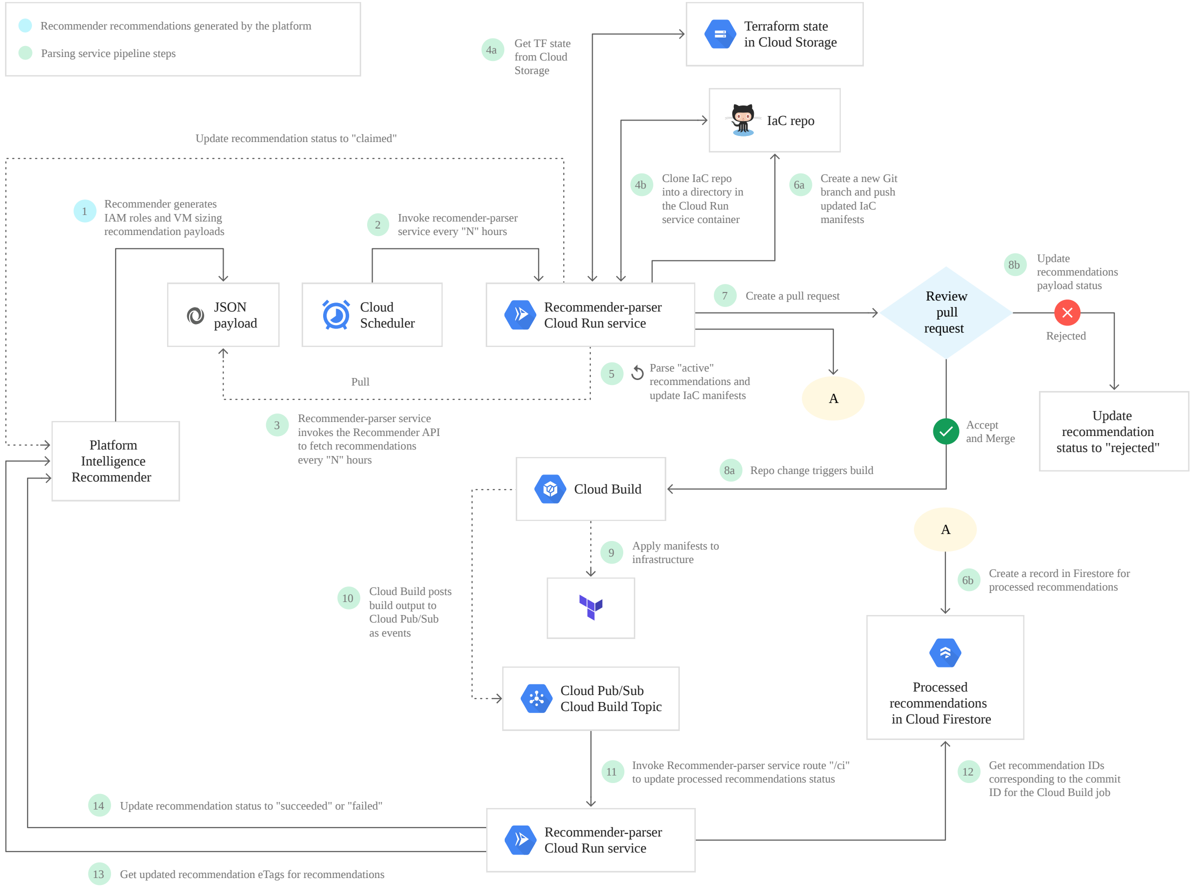
*An article about computational result is advertising, not scholarship. The actual scholarship is the **full software environment**, **code** and **data**, that produced the result.*

Cloud Platforms and Infrastructure as Code (IaC)

- Cloud platforms have APIs
- 😃 Setups of cloud infrastructure can be coded (IaC)
- 👍 Analysis code and cloud IaC on GitHub + Data → **Reproducibility**

Unfortunately, cloud infrastructure is mostly Do-It-Yourself (DIY) 😭

Infrastructure as Code



● Recommender recommendations generated by the platform

● Parsing service pipeline steps

Update recommendation status to "claimed"

1 Recommender generates IAM roles and VM sizing recommendation payloads

2 Invoke recomender-parser service every "N" hours

JSON payload

Cloud Scheduler

Recommender-parser Cloud Run service

Pull

3 Recommender-parser service invokes the Recommender API to fetch recommendations every "N" hours

Platform Intelligence Recommender

4a Get TF state from Cloud Storage

Terraform state in Cloud Storage

IaC repo

4b Clone IaC repo into a directory in the Cloud Run service container

6a Create a new Git branch and push updated IaC manifests

5 Parse "active" recommendations and update IaC manifests

A

7 Create a pull request

Review pull request

8b Update recommendations payload status

Rejected

Update recommendation status to "rejected"

Accept and Merge

8a Repo change triggers build

Cloud Build

9 Apply manifests to infrastructure

10 Cloud Build posts build output to Cloud Pub/Sub as events

Cloud Pub/Sub Cloud Build Topic

A

6b Create a record in Firestore for processed recommendations

Processed recommendations in Cloud Firestore

11 Invoke Recommender-parser service route "/ci" to update processed recommendations status

12 Get recommendation IDs corresponding to the commit ID for the Cloud Build job

Recommender-parser Cloud Run service

14 Update recommendation status to "succeeded" or "failed"

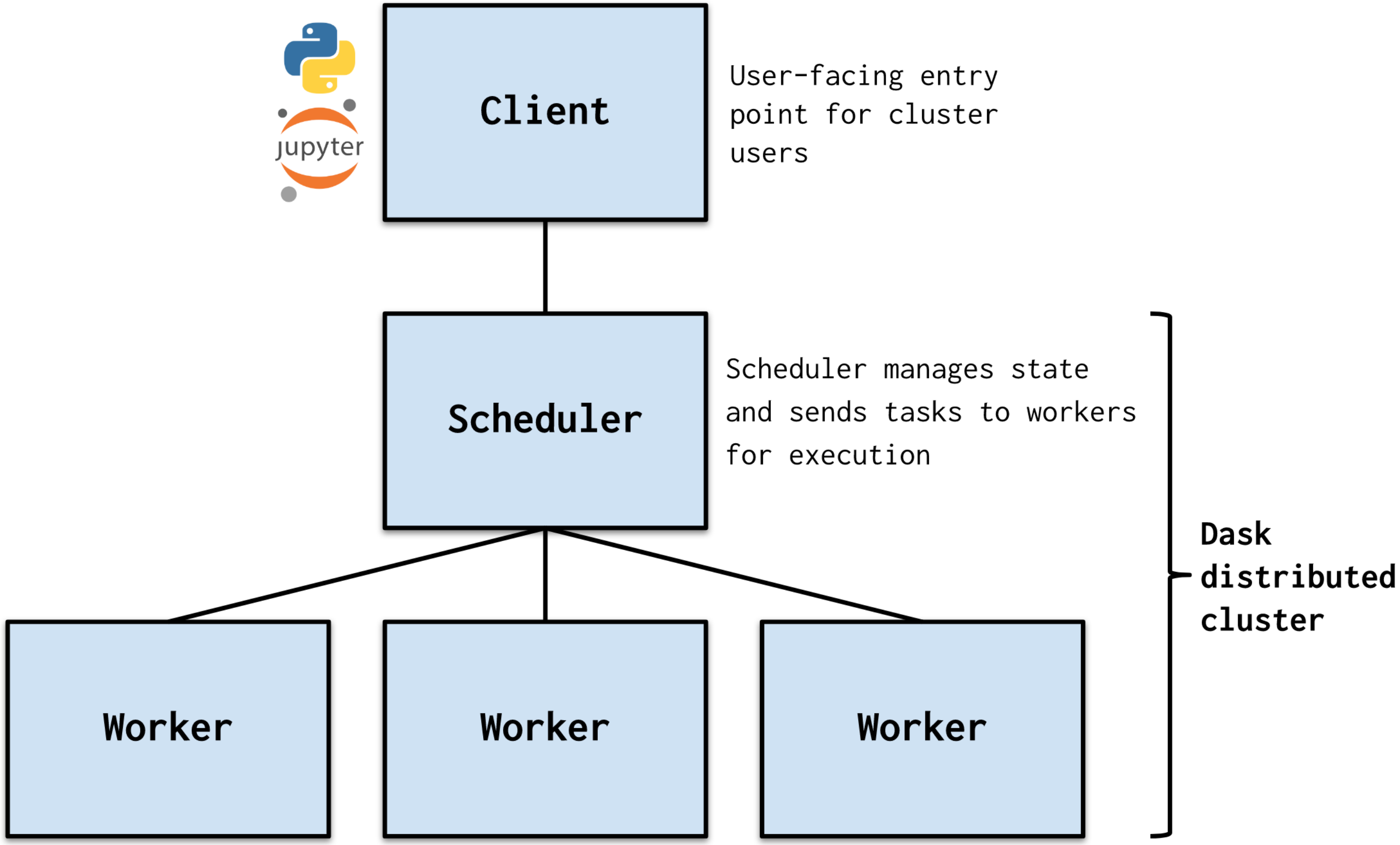
13 Get updated recommendation eTags for recommendations

(Using Recommendations for Infrastructure as Code)

Goal for Today

- Conceptual understanding of distributed computing infrastructure for Dask
- Discuss enabling technologies
- Hands-on lab using Google Cloud Platform

Dask Architecture



Client

User-facing entry point for cluster users

Scheduler

Scheduler manages state and sends tasks to workers for execution

Dask distributed cluster

Worker

Worker

Worker

Workers compute tasks / store and serve computed results to other workers or clients

Architecture Components

Client, scheduler, and worker processes can be distributed in different ways

- 🛖 *Before 2006*: physical server run (a combination of) processes
- 🏡 *Before 2015*: multiple virtual machines on powerful machines
- 🏨 *After 2015*: Kubernetes cluster to handle orchestration

What is Kubernetes? 🤷

Kubernetes and Containers

Kubernetes *is an open-source system for automating deployment, scaling, and management of containerized applications.*

Application containerization *(e.g. Docker) is an OS-level virtualization method used to deploy and run distributed applications without launching an entire virtual machine (VM) for each app.*

Enabling Technologies

docker	Component contents	HELM	Components wiring diagram
kubernetes	Builder and manager	Google Cloud	Provides resources

Hands-on Lab

Google Cloud Command Line Tool: https://shell.cloud.google.com/

Main tools

```
gcloud --version        # controls Google Cloud resources
docker --version        # container-level controls
kubectl version         # cluster-level controls
helm version            # installation "blueprint"
```

Login to Google Cloud

```
gcloud auth list                    # check current account
# gcloud auth login syoh@ucsb.edu   # login using another account

# set default project
gcloud config set project testing-sandbox-324502
gcloud config set compute/zone us-central1-a
```

Start Kubernetes Cluster

```
# create Kubernetes cluster
gcloud container clusters create \
    --machine-type e2-standard-4 \
    --num-nodes 2 \
    [unique-cluster-name]
```

- Machine type documentation
- Regions and zones documentation
- Use unique cluster names: e.g. include your initials

Dask Cluster Helm Chart

Three important concepts for Helm:

1. ***Chart***: a bundle of information necessary to create an instance of a Kubernetes application. (the blueprint)
2. ***Config***: configuration information that can be merged into a packaged chart. (user specified setting)
3. ***Release***: running instance of a *chart*, combined with *config*. (deployed instance)

Dask Helm Chart 🔗

Blueprint for setting up *Jupyter Lab*, *scheduler*, and *workers* on Kubernetes cluster

Dask Helm Chart Config

- User configuration supercedes default values in `values.yaml` file

```
cat << EOF > config.yaml
jupyter:
    serviceType: "LoadBalancer" # makes Jupyter notebook publicly accessible
scheduler:
    serviceType: "LoadBalancer" # makes Dask scheduler publicly accessible
worker:
    replicas: 4
EOF
```

- Instantiate Dask Cluster

```
helm repo add dask https://helm.dask.org/
helm repo update
helm install --wait my-dask -f config.yaml dask/dask    # takes a while
```

Kubernetes running `my-dask` release

```
sangoh@cloudshell:~ (testing-sandbox-324502)$ kubectl get all
NAME                                     READY   STATUS    RESTARTS   AGE
pod/my-dask-jupyter-54ddbfdd9d-rbsjb     1/1     Running   0          8m45s
pod/my-dask-scheduler-7f4f94bb7d-4c4fn   1/1     Running   0          8m45s
pod/my-dask-worker-6877d8f79f-42jrb      1/1     Running   0          8m44s
pod/my-dask-worker-6877d8f79f-88wvm      1/1     Running   1          8m45s
pod/my-dask-worker-6877d8f79f-9qvc5      1/1     Running   0          8m44s
pod/my-dask-worker-6877d8f79f-g659d      1/1     Running   1          8m45s
pod/my-dask-worker-6877d8f79f-htg8p      1/1     Running   1          8m45s
pod/my-dask-worker-6877d8f79f-t44jn      1/1     Running   1          8m45s

NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                       AGE
service/kubernetes          ClusterIP      10.12.0.1      <none>          443/TCP                       11m
service/my-dask-jupyter     LoadBalancer   10.12.8.243    34.121.167.99   80:32536/TCP                  8m45s
service/my-dask-scheduler   LoadBalancer   10.12.14.160   34.67.163.35    8786:30322/TCP,80:32005/TCP   8m45s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-dask-jupyter     1/1     1            1           8m45s
deployment.apps/my-dask-scheduler   1/1     1            1           8m45s
deployment.apps/my-dask-worker      6/6     6            6           8m45s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/my-dask-jupyter-54ddbfdd9d     1         1         1       8m46s
replicaset.apps/my-dask-scheduler-7f4f94bb7d   1         1         1       8m46s
replicaset.apps/my-dask-worker-6877d8f79f      6         6         6       8m46s
```

Google Cloud Kubernetes Workloads Overview 🔗

Server addresses

```
export DASK_SCHEDULER=$(kubectl get svc --namespace default my-dask-scheduler -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
export DASK_SCHEDULER_UI_IP=$(kubectl get svc --namespace default my-dask-scheduler -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
export DASK_SCHEDULER_PORT=8786
export DASK_SCHEDULER_UI_PORT=80

export JUPYTER_NOTEBOOK_IP=$(kubectl get svc --namespace default my-dask-jupyter -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
export JUPYTER_NOTEBOOK_PORT=80

echo tcp://$DASK_SCHEDULER:$DASK_SCHEDULER_PORT               -- Dask Client connection
echo http://$DASK_SCHEDULER_UI_IP:$DASK_SCHEDULER_UI_PORT     -- Dask dashboard
echo http://$JUPYTER_NOTEBOOK_IP:$JUPYTER_NOTEBOOK_PORT       -- Jupyter notebook
```

Run Example Notebook

1. Open Jupyter notebook specified in output
2. Copy dashboard URL into Dask Jupyter lab extension
3. Open and run `examples/04-dask-array.ipynb`
4. Additional packages are needed for `examples/05-nyc-taxi.ipynb`

Install Additional Packages

```
cat << EOF > config.yaml
jupyter:
  serviceType: "LoadBalancer" # makes Jupyter notebook publicly accessible
  env:
    - name: EXTRA_PIP_PACKAGES
      value: "pyarrow gcsfs"

scheduler:
  serviceType: "LoadBalancer" # makes Dask scheduler publicly accessible

worker:
  replicas: 4
  env:
    - name: EXTRA_PIP_PACKAGES
      value: "pyarrow gcsfs"

EOF
```

Upgrade `my-dask` Release

- `upgrade` release rather than `install`

  ```
  helm upgrade --wait my-dask -f config.yaml dask/dask    # takes a while
  ```

- Run `examples/05-nyc-taxi.ipynb` (will break). Check why with `kubectl`

  ```
  kubectl get all                                 # what do you notice?
  kubectl describe [pod/my-dask-worker-00000]     # evicted resource name
  ```

Try again

```
cat << EOF > config.yaml
jupyter:
  serviceType: "LoadBalancer" # makes Jupyter notebook publicly accessible
  resources:
    limits:
      cpu: 1
      memory: 3G
    requests:
      cpu: 0.5
      memory: 2G
  env:
    - name: EXTRA_PIP_PACKAGES
      value: "pyarrow gcsfs matplotlib"

scheduler:
  serviceType: "LoadBalancer" # makes Dask scheduler publicly accessible
  resources:
    limits:
      cpu: 1
      memory: 3G
    requests:
      cpu: 0.5
      memory: 2G

worker:
  replicas: 15
  resources:
    limits:
      cpu: 1
      memory: 3G
    requests:
      cpu: 0.5
      memory: 2.5G
  env:
    - name: EXTRA_PIP_PACKAGES
      value: "pyarrow gcsfs"

EOF
```

Upgrade `my-dask` Release again

- Resize cluster to add nodes

```
gcloud container clusters resize [your-cluster-name] --num-nodes 5
```

- `upgrade` release rather than `install`

```
helm upgrade --wait my-dask -f config.yaml dask/dask    # takes a while
```

- Run `examples/05-nyc-taxi.ipynb`