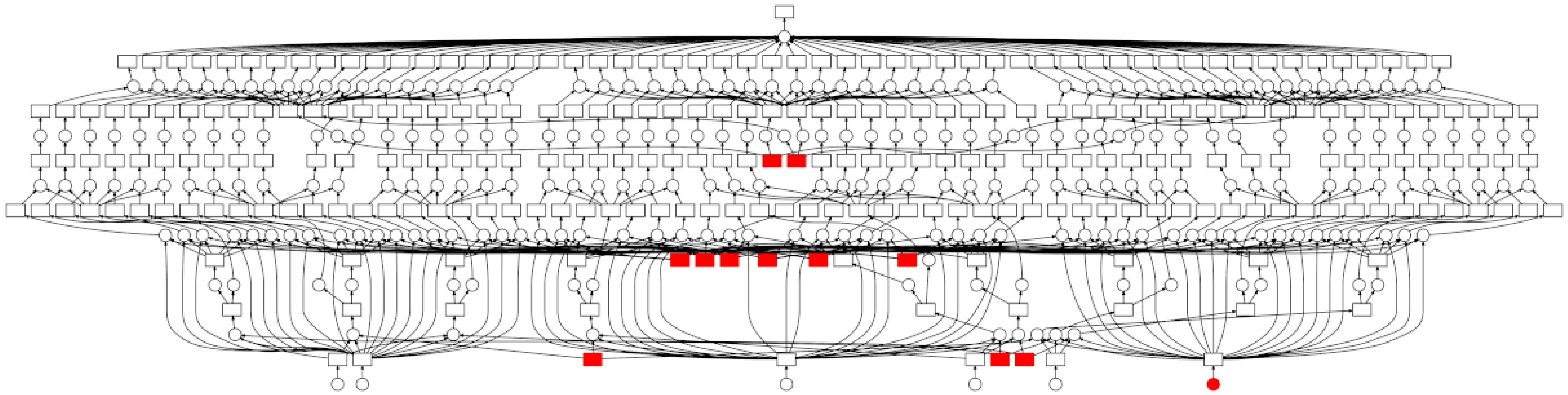
Dask and V100s for Fast, Distributed Batch Scoring of Computer Vision Workloads

Mathew Salvaris

- What is Dask?
- Batch scoring use cases
  - Style transfer
  - Mask- RCNN for object detection and segmentation

DASK



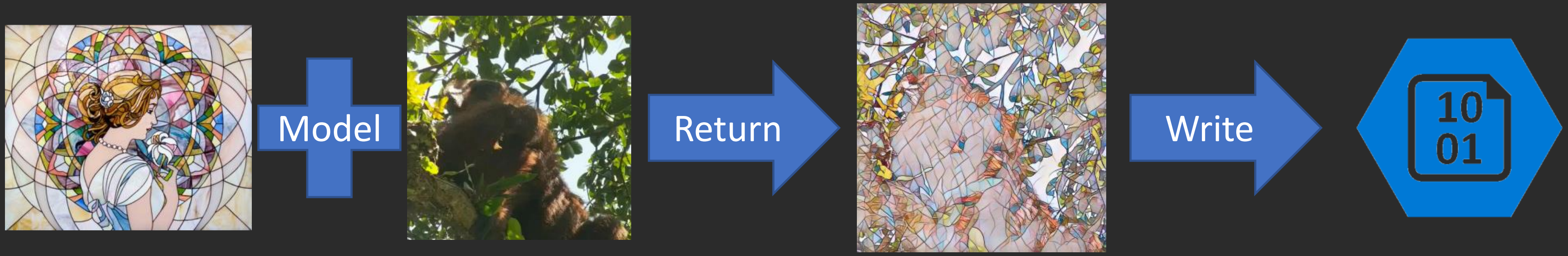
http://docs.dask.org/en/latest/why.html

Dask for batch scoring

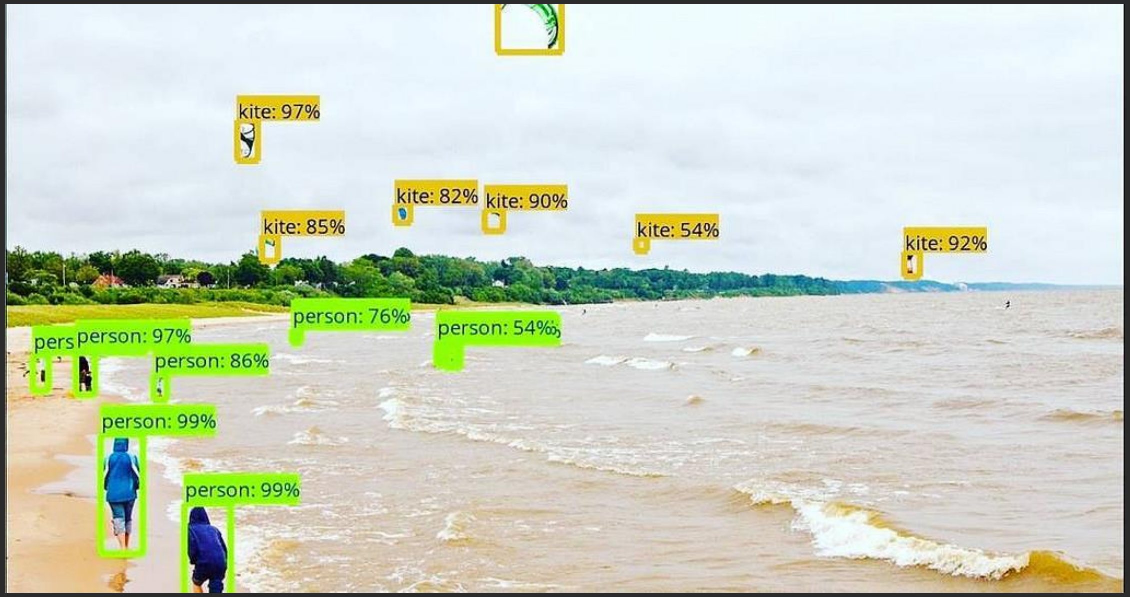
- Same code runs on single node locally or in a cluster
- No orchestration code to write, Dask handles orchestration
- Can create and execute complex DAGs that can be generated on the fly

Batch scoring use cases

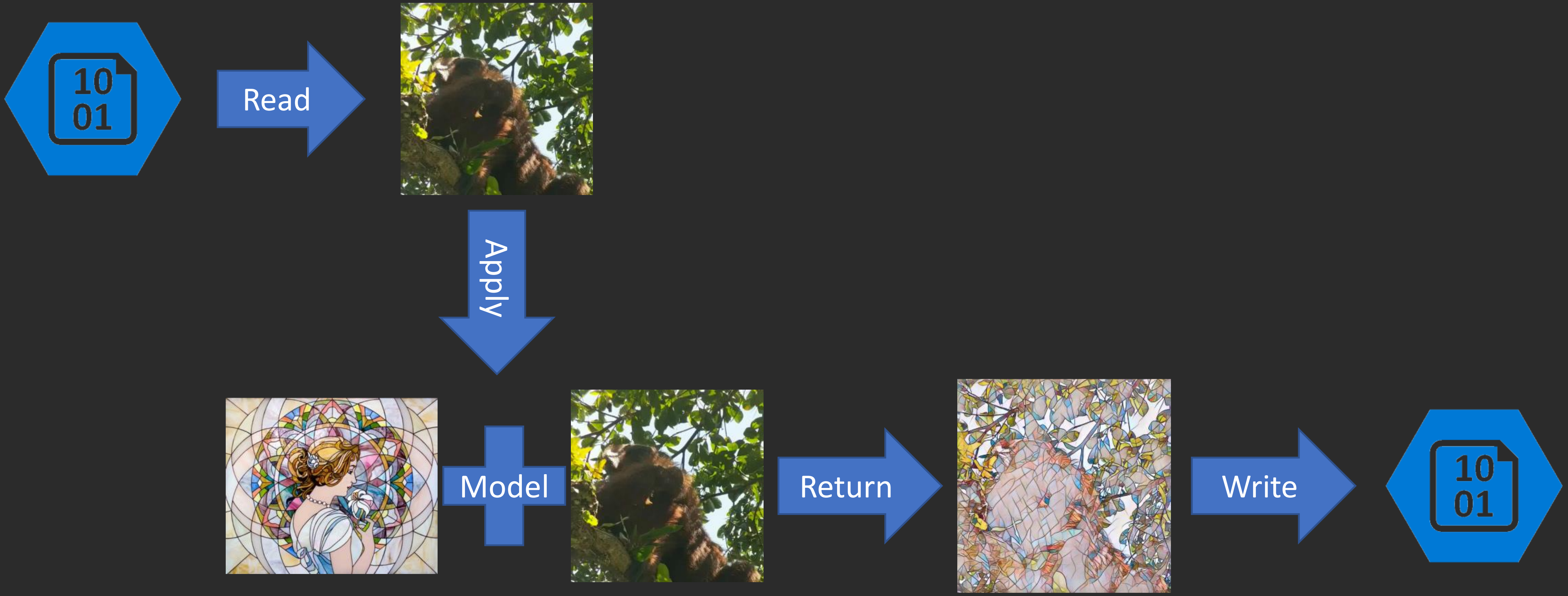
- Style Transfer



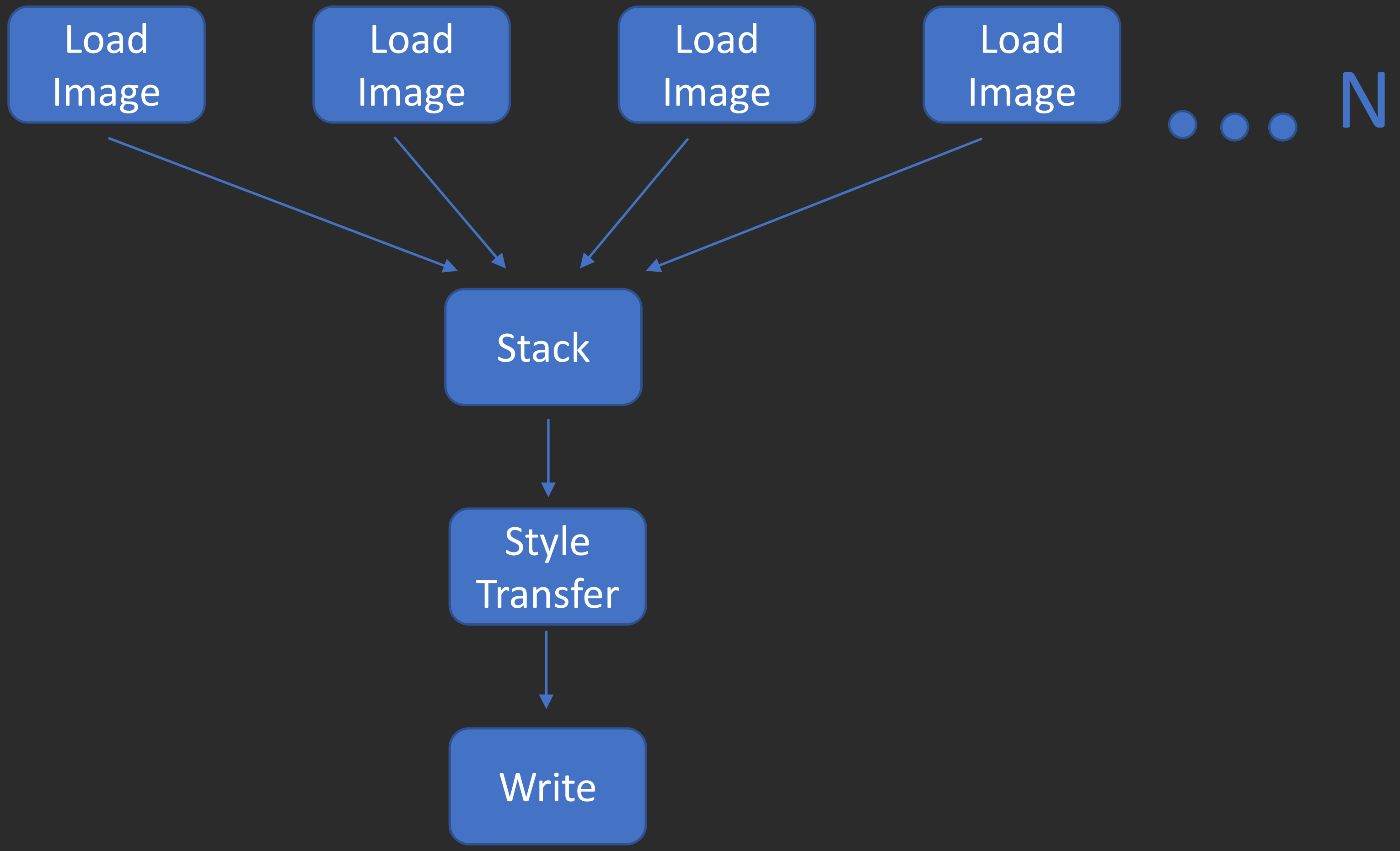
- Object detection and segmentation



Style transfer process



Dask Graph of Style Transfer Process



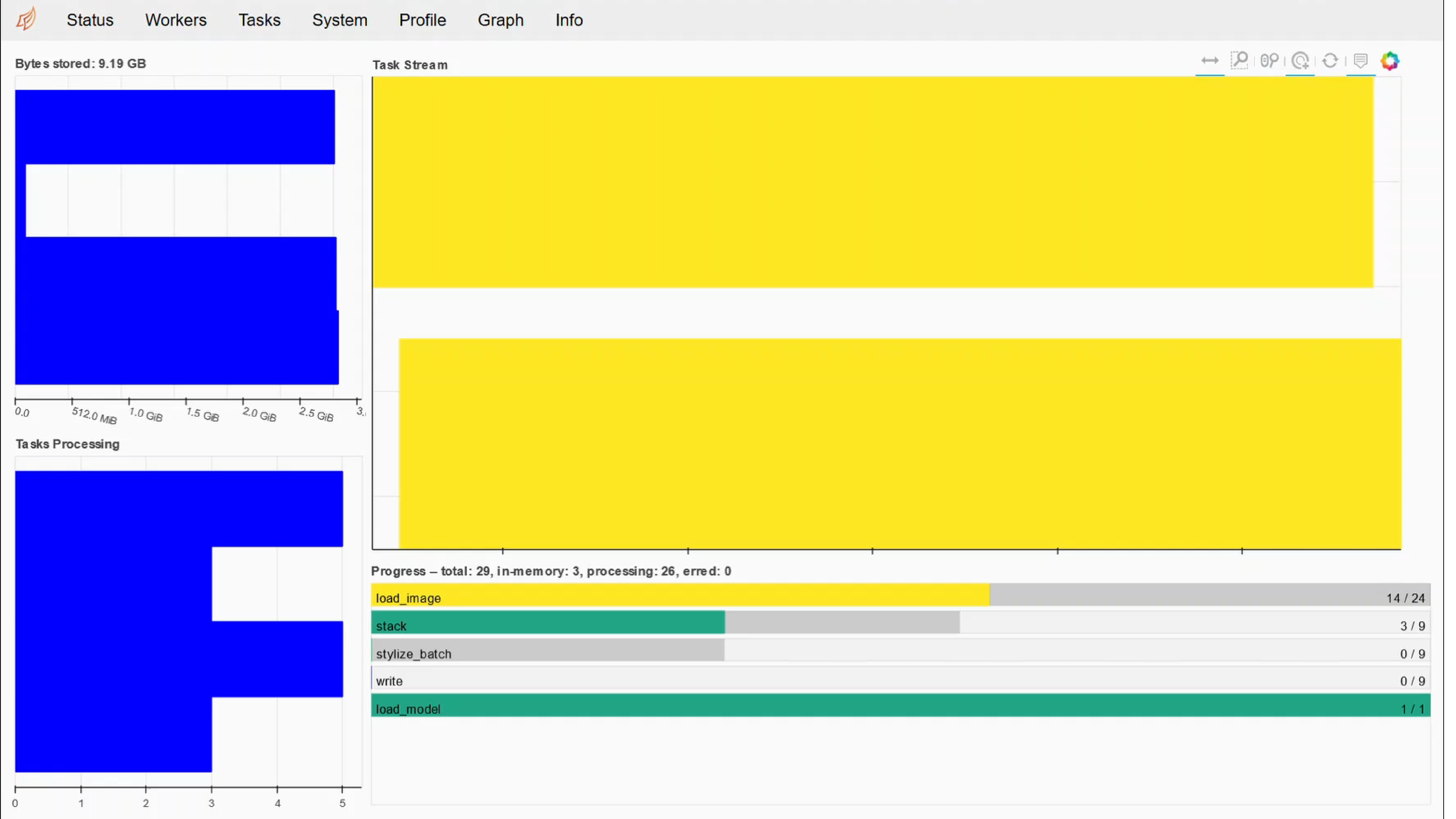
```python
@curry
def process_batch(client, style_model, output_path, batch_filenames):
    remote_batch_f = client.scatter(batch_filenames)
    img_array_f = client.map(load_image, remote_batch_f)
    stacked_array_f = client.submit(stack, img_array_f)
    styled_array_f = client.submit(stylize_batch, style_model, stacked_array_f)
    return client.submit(write, batch, styled_array_f, output_path)
```

Testing Dask locally on GPU

- Need GPU based libraries for workers and CPU based ones client
- Would limit the visibility of CUDA devices to each worker

```
CUDA_VISIBLE_DEVICES=0 dask-worker 127.0.0.1 --nprocs 1 --nthreads 1 --resources 'GPU=1
```

- Spawn as many workers as there are GPUs

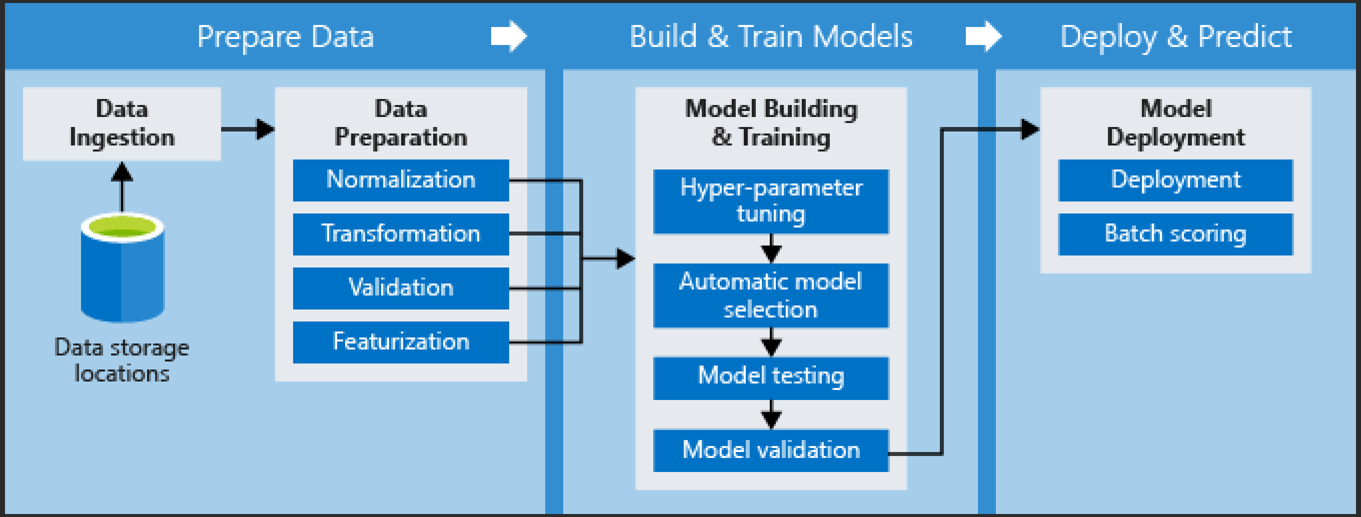


Progress – total: 29, in-memory: 3, processing: 26, erred: 0

```
load_image        14 / 24
stack              3 / 9
stylize_batch      0 / 9
write              0 / 9
load_model         1 / 1
```

What is Azure ML Pipelines

Create experimentation graph that gets you from data to a model

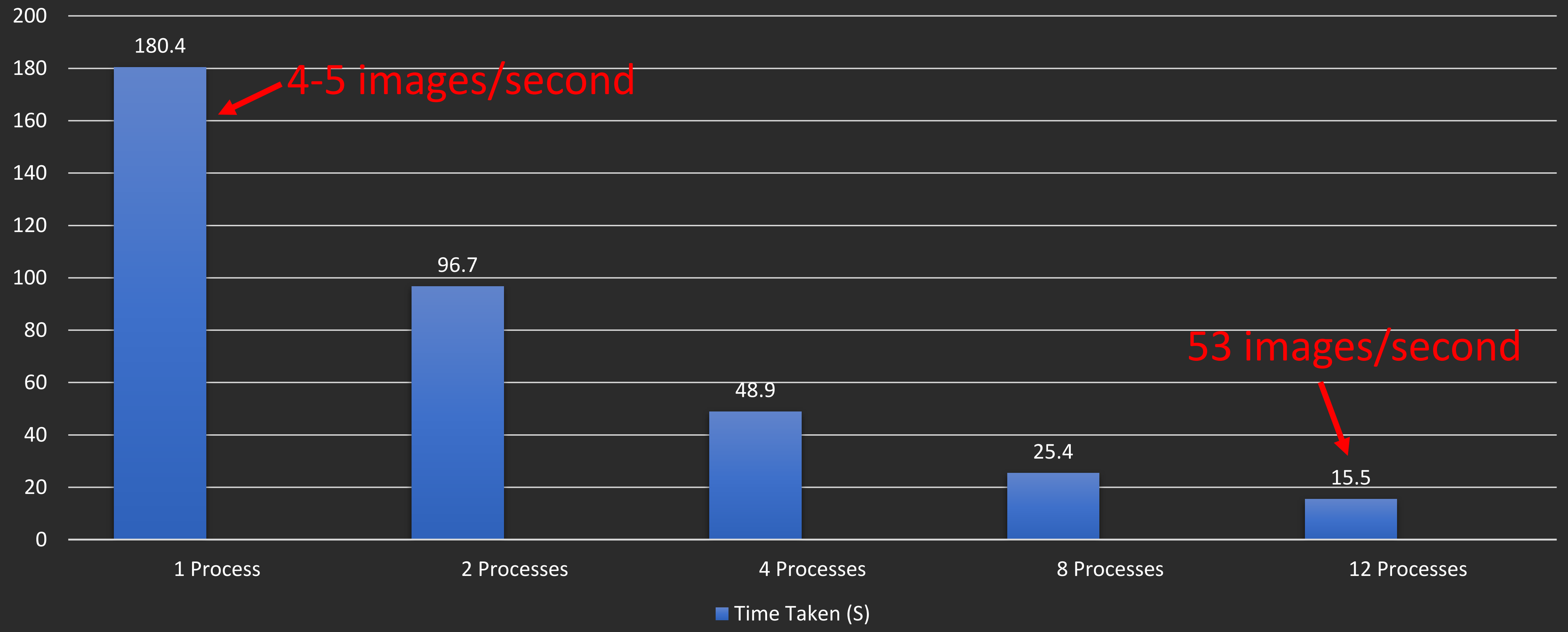


The pipeline can be exported as a parameterised end point

Dask on Azure pipelines

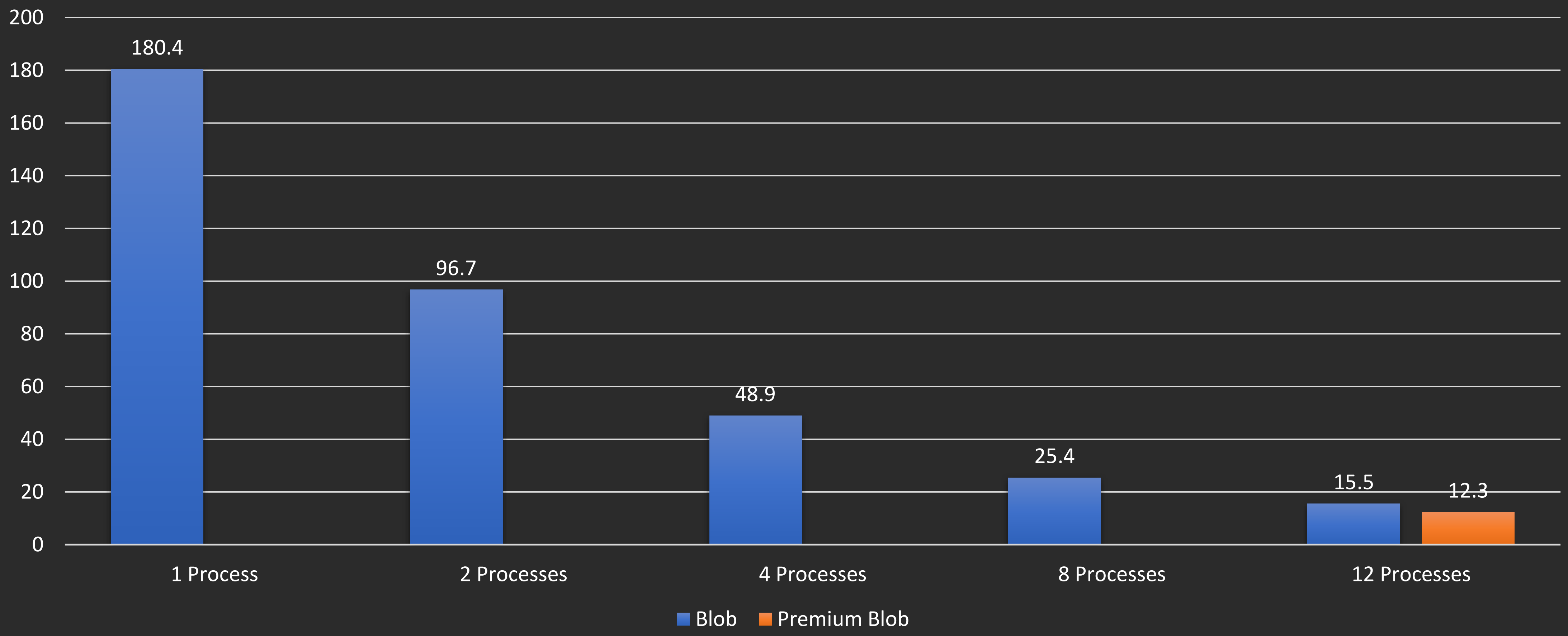
- Need to set off workers, scheduler and run client
- Azure ML pipelines has MPIStep which allows us to trigger MPI job

- Run workers on all ranks - Run client and scheduler on rank 0



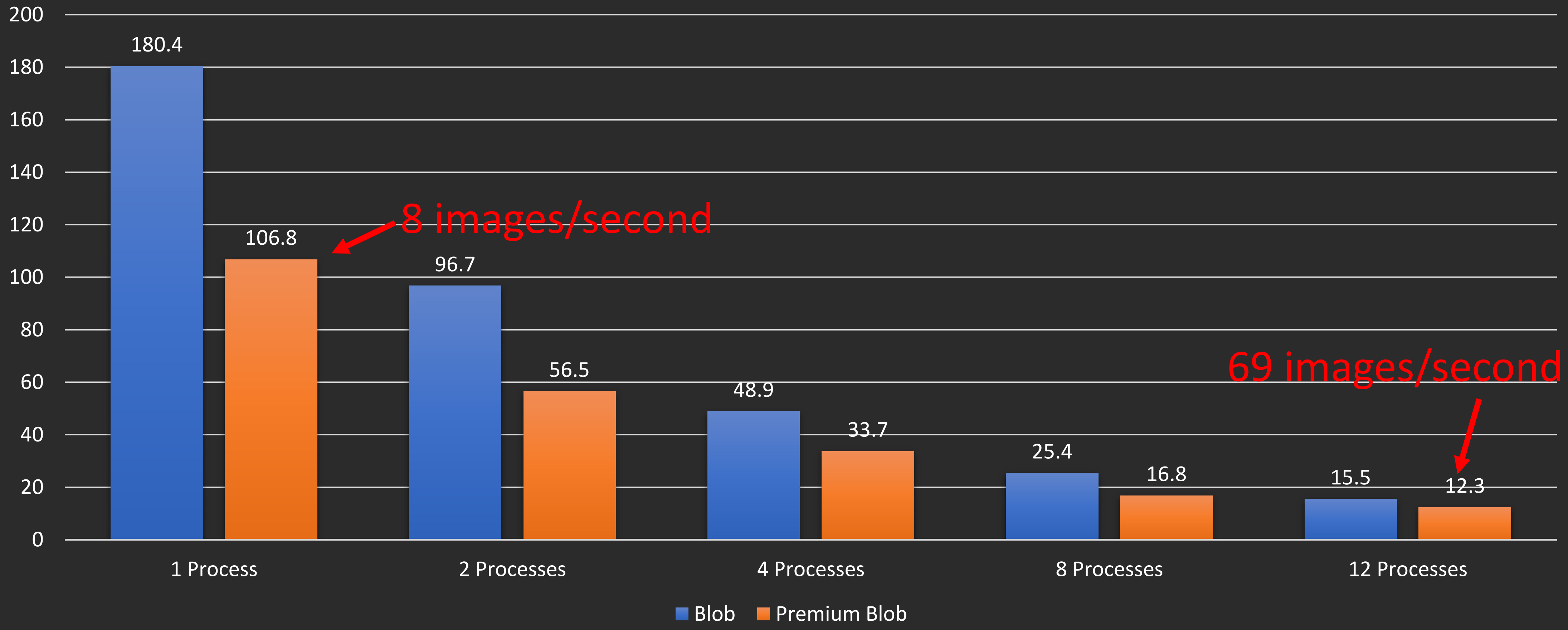
GPU: V100

# Images: 823



GPU: V100

# Images: 823

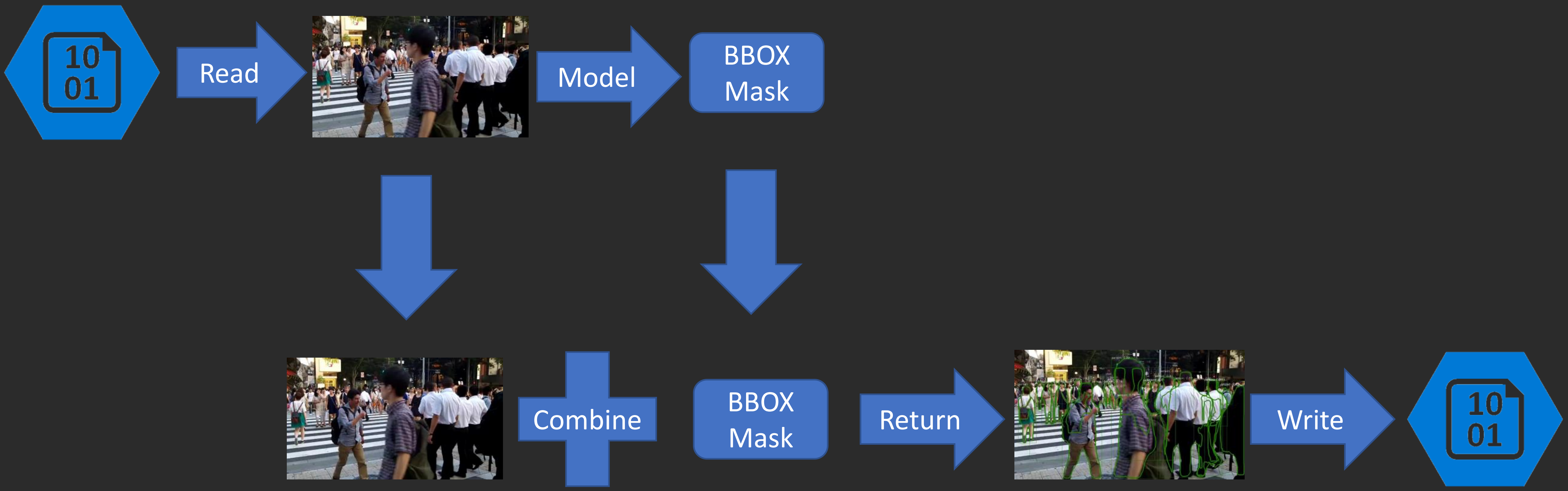


GPU: V100

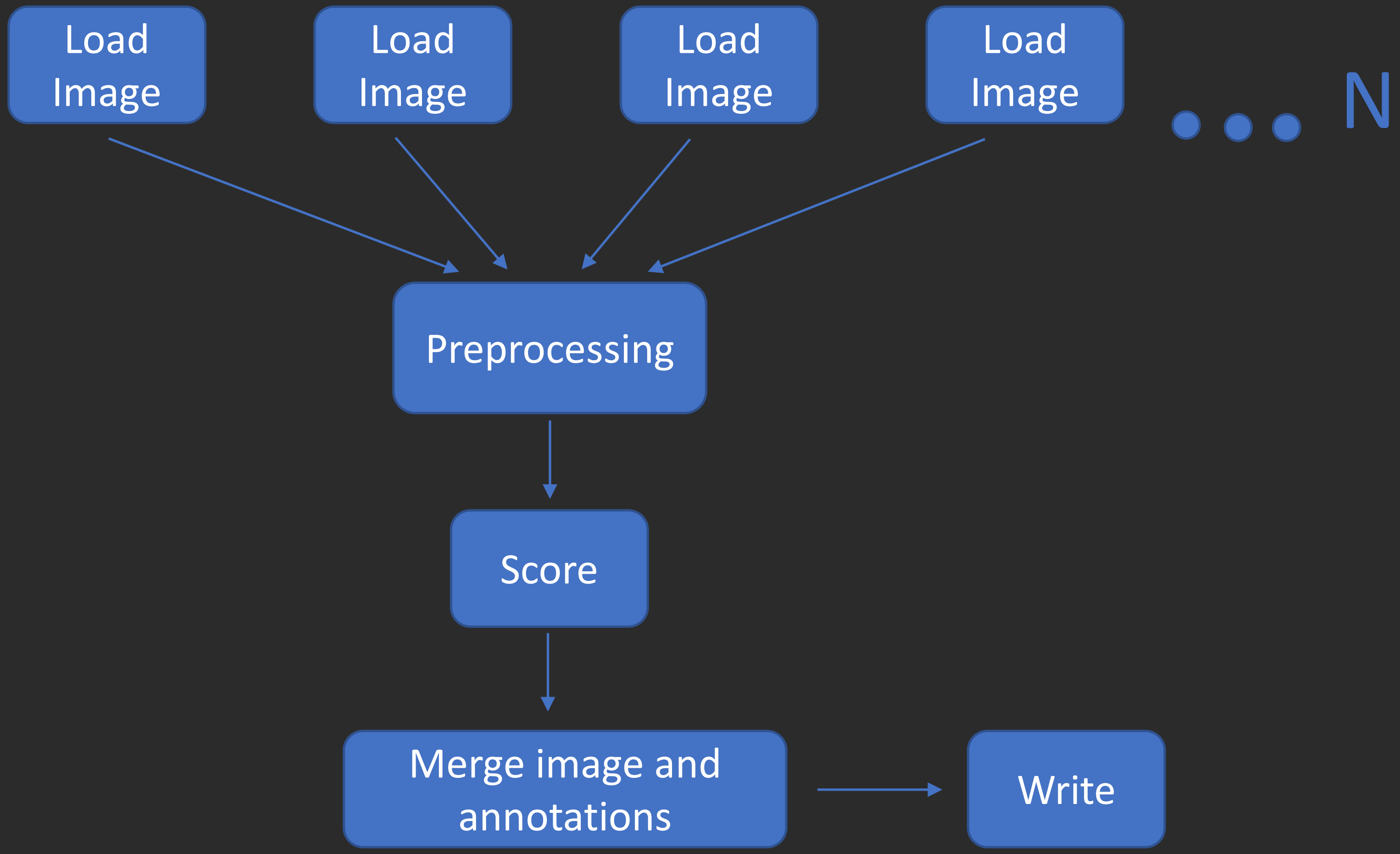
# Images: 823

Azure ML Pipelines Notebook

Mask-RCNN process



Dask Graph of Mask-RCNN



```python
@curry
def process_batch(client, style_model, preprocessing, output_path, batch):
    remote_batch_f = client.scatter(batch)
    img_array_f = client.map(load_image, remote_batch_f)
    pre_img_array_f = client.map(preprocessing, img_array_f)
    bbox_list_f = client.submit(score_batch, style_model, pre_img_array_f)
    results_f = client.submit(loop_annotations, img_array_f, bbox_list_f)
    return client.submit(loop_write(output_path), batch, results_f)
```

Dask on Kubernetes

- Use Helm chart to deploy, workers, scheduler and Jupyter lab
- Provision 3 VMs with 4 V100s each
- Installed Nvidia device plugin
- Installed plugin for storage

Demo Kubernetes

Summary

✓ Able to easily prototype two batch scoring scenarios locally then deploy on Kubernetes as well as Azure pipelines

Using GPUs through Dask could be more straight forward, better interaction between Dask and DL libraries

Acknowledgements

JS Tan

Azure Machine Learning Pipelines Team

Thanks
&
Questions?