My first Azure Batch AI Job

January 15, 2018

Azure Batch AI provides us the PaaS opportunity to use GPU resources in the cloud. The basis is to use virt
(i.e. you don't have to maintain them) and run jobs as you see fit. For my use case, the opportunity of low-p
using GPU machines is also particularly promising.

What I'll run through is running our first job on Azure Batch AI. For setup, we'll use the Azure CLI as I find it
portal. Saying that, everything can be achieved by point and click at portal.azure.com

Assuming you already have the CLI installed and you are already logged in with it.

Resource Group

As with all Azure resources, we group them with a resource group and you'll have to know your desired loca
machine, this restricts you a bit, but you can find which regions are supported at https://azure.microsoft.co
for Virtual Machines of NC, NCv2, NCv3.

For my purposes, my resource group will be `batch-rg` and will be in location westus2.

```
# Configure CLI defaults so we don't have to pass them to all commands
az configure --defaults group='batch-rg' location='westus2'

# Create our resource group
az group create -n 'batch-rg'
```

Storage

Azure storage is used for accessing your code and will be used to read in data and write the model checkp

Setting a few environment variables also simplifies CLI usage so we create an account named `pontifexml` a
as you'll see below.

```
az storage account create \
    -n pontifexml \
    --sku Standard_LRS

# Set our storage account name for CLI and Batch AI CLI
export {AZURE_BATCHAI_STORAGE_ACCOUNT,AZURE_STORAGE_ACCOUNT}=pontifexml

# Set our storage account key for CLI and Batch AI CLI
export {AZURE_BATCHAI_STORAGE_KEY,AZURE_STORAGE_KEY}=$(az storage account keys list --acco
```

We will also create a file share that the cluster will use named `machinelearning`:

```
az storage share create \
    -n machinelearning
```

Batch AI Cluster

```
az batchai cluster show -n dsvm -o table
```

Batch AI Job

To ensure our cluster is working correctly, our job will be a TensorFlow hello world in a file ‘hello-tf.py’ with

```
"""
TensorFlow 'Hello World'

Author: Damien Pontifex
"""

import tensorflow as tf

def run_training():
    """Run a 'training' sample"""

    hello = tf.constant('Hello, TensorFlow!')

    with tf.Session() as sess:
        print(sess.run(hello))

if __name__ == '__main__':
    run_training()
```

We need this code to be in our Azure file share and we will upload it using the CLI:

```
# Create a directory in our share
az storage directory create \
    --share-name machinelearning \
    --name helloworld

# Upload our hello-tf.py file to that directory
az storage file upload \
    --share-name machinelearning \
    --path helloworld \
    --source hello-tf.py

# Upload any data files
```

Now to define how our job should run, we use a JSON file which for this job is: (job.json)

```
{
    "properties": {
        "nodeCount": 1,
        "tensorFlowSettings": {
            "pythonScriptFilePath": "$AZ_BATCHAI_INPUT_SCRIPT/hello-tf.py",
            "masterCommandLineArgs": "-p"
        },
        "stdOutErrPathPrefix": "$AZ_BATCHAI_MOUNT_ROOT/afs/helloworld",
        "inputDirectories": [
            {
                "id": "SCRIPT",
                "path": "$AZ_BATCHAI_MOUNT_ROOT/afs/helloworld"
            }
        ],
```

Once completed you should see the stdout of ‘Hello, TensorFlow!’ in the fileshare.

Conclusion

Even though this is a ‘hello world’ example, the only change to this workflow for an actual ML job would be
script. From here we could run any job we like.

As a side note I found, if you’re using Jupyter notebooks for development locally, you can run these via the
`nbconvert --to notebook --execute mynotebook.ipynb` and the notebook will be executed similar to a R
make sure to use the environment variables as appropriate for any data locations