Aer_Distrubited_with_Kubernetes_and_more

September 17, 2021

[1]:
```
import numpy as np
```

[2]:
```
from qiskit import QuantumCircuit
```

[3]:
```
from qiskit import Aer
```

[4]:
```
from qiskit.tools.visualization import plot_histogram, plot_state_city
```

[5]:
```
Aer.backends()
```

[5]:
```
[AerSimulator('aer_simulator'),
 AerSimulator('aer_simulator_statevector'),
 AerSimulator('aer_simulator_density_matrix'),
 AerSimulator('aer_simulator_stabilizer'),
 AerSimulator('aer_simulator_matrix_product_state'),
 AerSimulator('aer_simulator_extended_stabilizer'),
 AerSimulator('aer_simulator_unitary'),
 AerSimulator('aer_simulator_superop'),
 QasmSimulator('qasm_simulator'),
 StatevectorSimulator('statevector_simulator'),
 UnitarySimulator('unitary_simulator'),
 PulseSimulator('pulse_simulator')]
```

[6]:
```
simulator=Aer.get_backend('aer_simulator')
```

[7]:
```
from qiskit.circuit.random import random_circuit
```

[8]:
```
qc=[random_circuit(num_qubits=3, depth=4, measure=True) for _ in range(1,11)]
```

[9]:
```
from qiskit import transpile
```

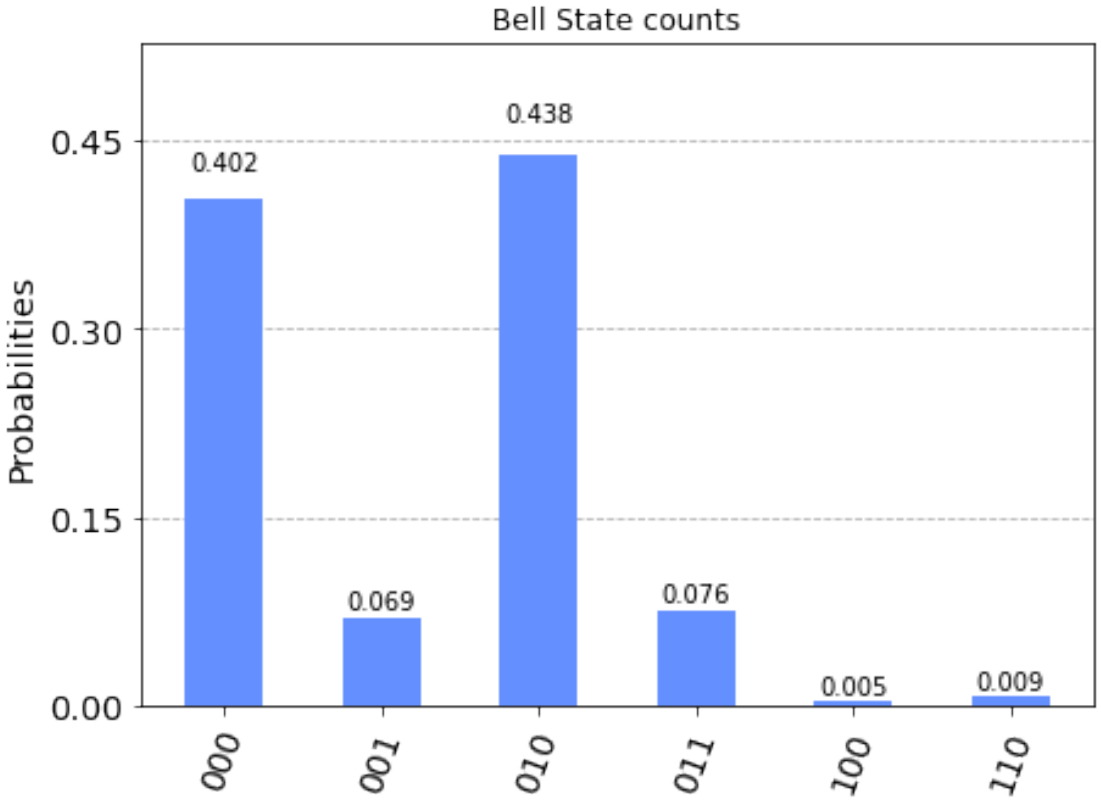
[10]:
```
qc=transpile(qc, simulator)
```

[11]:
```
result=simulator.run(qc).result()
```

[12]:
```
counts=result.get_counts(qc[0])
```

[13]:
```
plot_histogram(counts, title="Bell State counts")
```

[13]:



[ ]:
```
##############################################################################################################
```

[14]:
```
from dask_kubernetes import KubeCluster
```

[15]:
```
!cat ./worker-spec.yml
```

```
# worker-spec.yml

kind: Pod
metadata:
  labels:
    foo: bar
spec:
  restartPolicy: Never
  containers:
  - image: daskdev/dask:latest
    imagePullPolicy: IfNotPresent
    args: [dask-worker, --nthreads, '1', --no-dashboard, --memory-limit, 1G,
--death-timeout, '60']
```

```
    name: dask
    env:
      - name: EXTRA_PIP_PACKAGES
        value: git+https://github.com/dask/distributed
    resources:
      limits:
        cpu: "1"
        memory: 1G
      requests:
        cpu: "1"
        memory: 1G
```

[16]:
```
cluster_kube = KubeCluster('worker-spec.yml')
```

```
Creating scheduler pod on cluster. This may take some time.
Forwarding from 127.0.0.1:64978 -> 8786
Forwarding from [::1]:64978 -> 8786
Handling connection for 64978
Handling connection for 64978
Handling connection for 64978
```

```
/home/red/.local/lib/python3.9/site-packages/distributed/client.py:1100:
VersionMismatchWarning: Mismatched versions found

+-------------+-----------+-----------------------+---------+
| Package     | client    | scheduler             | workers |
+-------------+-----------+-----------------------+---------+
| blosc       | None      | 1.10.2                | None    |
| cloudpickle | 2.0.0     | 1.6.0                 | None    |
| distributed | 2021.09.0 | 2021.09.0+18.g05677bb2 | None    |
| lz4         | None      | 3.1.3                 | None    |
+-------------+-----------+-----------------------+---------+
  warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"]))
```

[17]:
```
cluster_kube.get_logs()
```

```
Handling connection for 64978
```

[17]:
```
{'Cluster': 'Creating scheduler pod on cluster. This may take some time.',
 'Scheduler': 'distributed.scheduler - INFO -
-----------------------------------------------\ndistributed.scheduler - INFO -
Clear task state\ndistributed.scheduler - INFO -   Scheduler at:
tcp://172.17.0.3:8786\ndistributed.scheduler - INFO -   dashboard at:
:8787\ndistributed.scheduler - INFO - Receive client connection: Client-
ba999591-1751-11ec-9a51-a09f10d41eae\ndistributed.scheduler - INFO - Remove
client Client-ba999591-1751-11ec-9a51-a09f10d41eae\ndistributed.scheduler - INFO
- Remove client Client-
ba999591-1751-11ec-9a51-a09f10d41eae\ndistributed.scheduler - INFO - Close
```

```
client connection: Client-ba999591-1751-11ec-9a51-a09f10d41eae'}
```

[ ]:
```
####   It dosen't run after this so moving on to next ways of running Aer
```

[ ]:
```
from dask.distributed import Client
```

[ ]:
```
client_kube = Client(cluster_kube)
```

[ ]:
```
qbackend = Aer.get_backend('qasm_simulator')
```

[ ]:
```
from qiskit import execute
```

[ ]:
```
result_ideal = execute(qc, qbackend, executor=client_kube).result()
```

[ ]:
```
counts=result.get_counts(qc[0])
```

[ ]:
```
plot_histogram(counts, title="Bell State counts")
```

[ ]:
```

```

[ ]:
```
client_kube.close()
```

[ ]:
```
cluster_kube.close()
```

[ ]:
```

```

[ ]:
```
##########################################################################################################
```

[ ]:
```

```

[ ]:
```

```

[18]:
```
from concurrent.futures import ThreadPoolExecutor
```

[19]:
```
exc_threadpool = ThreadPoolExecutor(max_workers=2)
```

[20]:
```
exc_threadpool
```

[20]:
```
<concurrent.futures.thread.ThreadPoolExecutor at 0x7fcffb7d41c0>
```

[21]:
```
qbackend = Aer.get_backend('qasm_simulator')
```

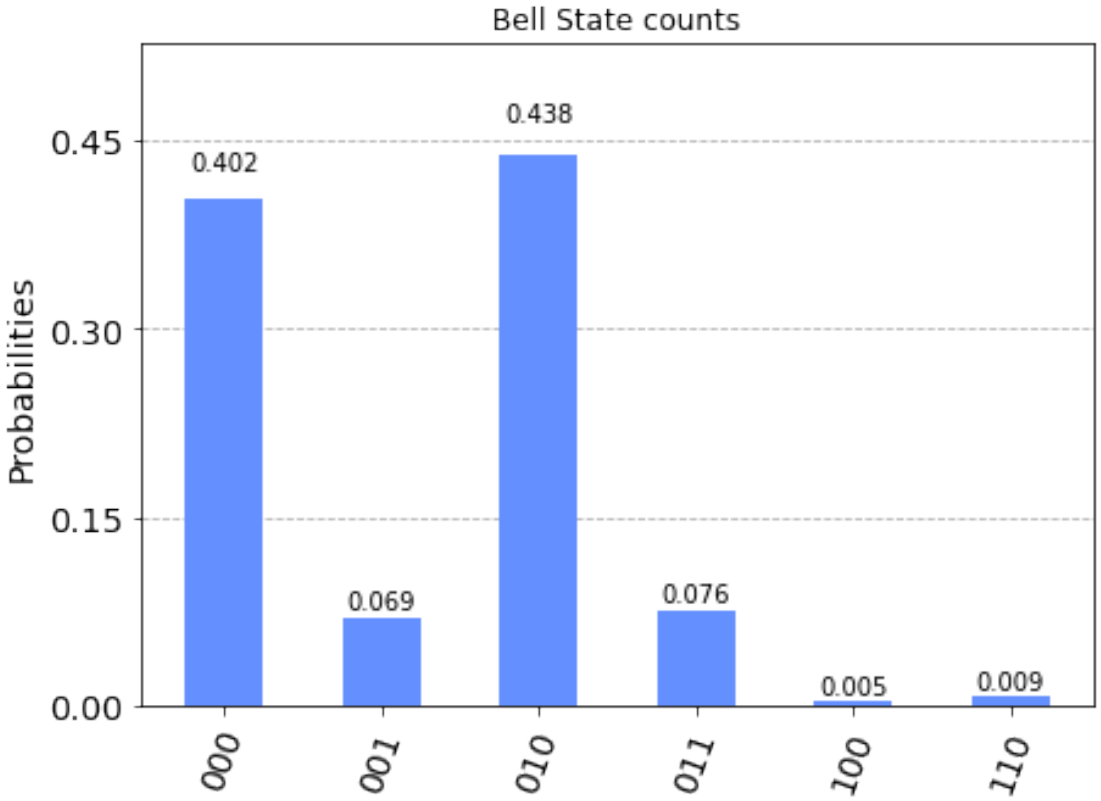
[22]:
```
from qiskit import execute
```

[23]:
```
result_ideal = execute(qc, qbackend, executor=exc_threadpool).result()
```

[24]:
```
counts=result.get_counts(qc[0])
```

[25]:
```
plot_histogram(counts, title="Bell State counts")
```

[25]:



[ ]:

[ ]:
```
##############################################################################################
```

[ ]:

[ ]:

[26]:
```
from dask.distributed import Client
```

[27]:
```
from dask.distributed import LocalCluster
```

[28]:
```
local_cluster_1=LocalCluster(n_workers=1, processes=True)
```

[29]:
```
local_cluster_1
```

```
Tab(children=(HTML(value='<div class="jp-RenderedHTMLCommon jp-RenderedHTML␣
↪jp-mod-trusted jp-OutputArea-outpu…
```

[30]:
```
client_localcluster = Client(address=local_cluster_1)
```

[31]:
```
client_localcluster
```

[31]:
```
<Client: 'tcp://127.0.0.1:39575' processes=1 threads=16, memory=7.68 GiB>
```

[32]:
```
qbackend = Aer.get_backend('qasm_simulator')
```

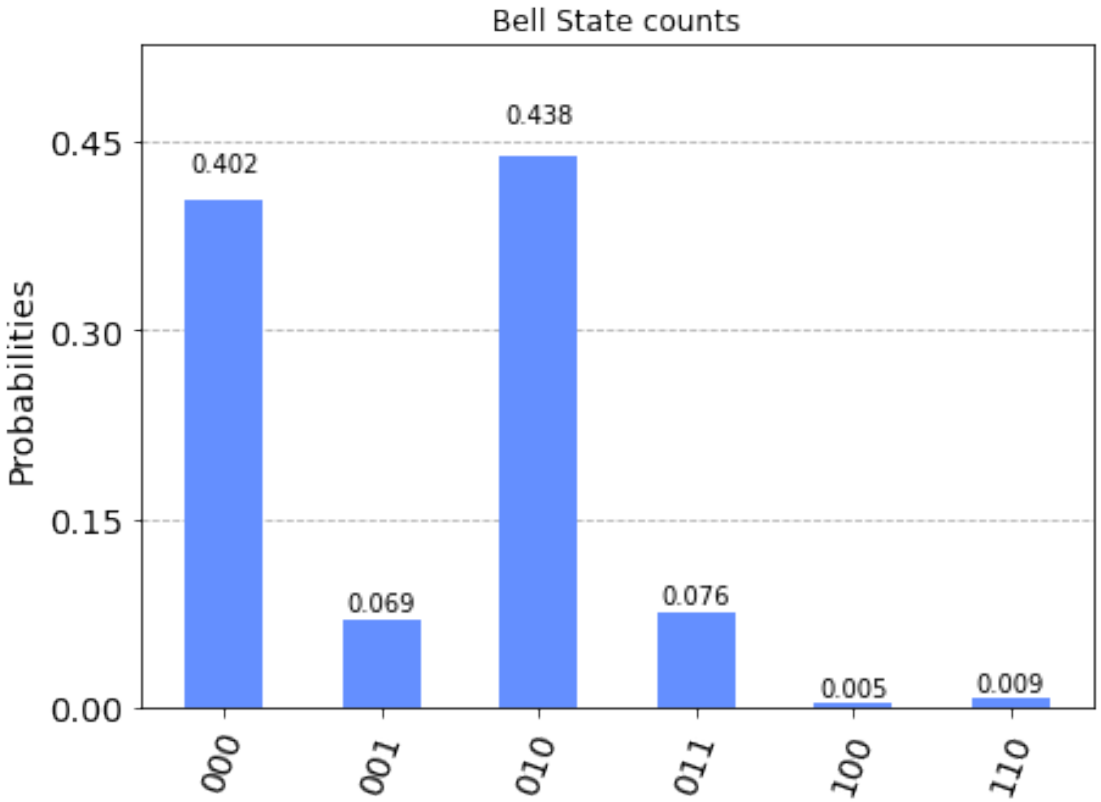
[33]:
```
from qiskit import execute
```

[34]:
```
result_ideal = execute(qc, qbackend, executor=client_localcluster).result()
```

[35]:
```
counts=result.get_counts(qc[0])
```

[36]:
```
plot_histogram(counts, title="Bell State counts")
```

[36]:



[37]:
```
local_cluster_1.close()
```

[38]:
```
client_localcluster.close()
```

[ ]:
```

```

[ ]:

[ ]: ##

[ ]:

[39]:
```
from dask.distributed import Client
```

[40]:
```
exc=Client(n_workers=2, threads_per_worker=1, memory_limit='500MB')
```

[41]:
```
qbackend = Aer.get_backend('qasm_simulator')
```

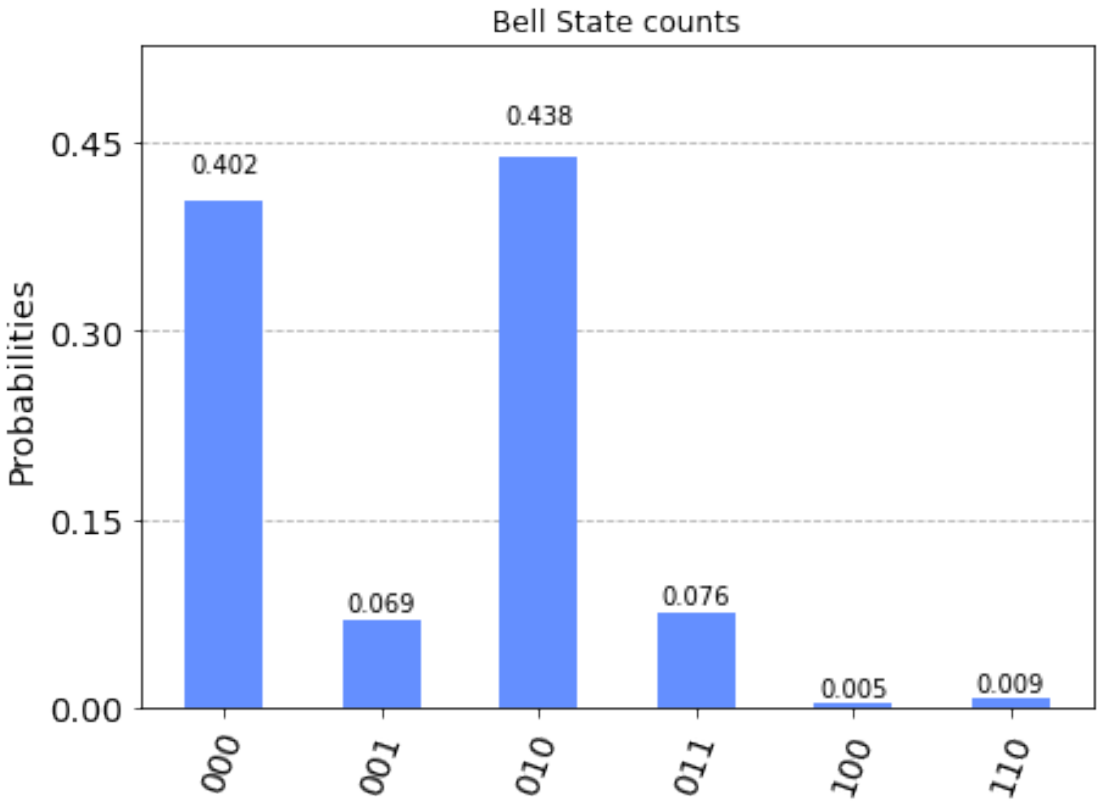
[42]:
```
from qiskit import execute
```

[43]:
```
result_ideal = execute(qc, qbackend, executor=exc).result()
```

[44]:
```
counts=result.get_counts(qc[0])
```

[45]:
```
plot_histogram(counts, title="Bell State counts")
```

[45]:



[46]:
```
exc.close()
```

[ ]:

[ ]: