# Ingest points in parallel into a TileDB array - 131,290,826 points

In [15]:

```
import boto3
import numpy as np
from tiledb.cloud.compute import Delayed
import tiledb
```

In [16]:

```
input_bucket = 's3://lidar-backend/ingest/'
create_array_uri = 'tiledb://norman/s3://lidar-backend/nj_ingest'
array_uri = 'tiledb://norman/nj_ingest'
```

In [17]:

```
# clean up previous runs
try:
    from tiledb.cloud import deregister_array
    deregister_array(array_uri, async_req=False)
except Exception as e:
    print(e)

!aws s3 rm s3://lidar-backend/nj_ingest --recursive --quiet
```

In [18]:

```
def get_aws_config():
    session = boto3._get_default_session()
    credentials = session.get_credentials()
    return dict(
        region_name=session.region_name,
        aws_access_key_id=credentials.access_key,
        aws_secret_access_key=credentials.secret_key,
        aws_session_token=credentials.token,
    )
```

In [19]:

```
inputs = []

conn = boto3.client('s3')
for key in conn.list_objects(Bucket='lidar-backend', Prefix='ingest')['Contents']:
    if key['Key'].endswith('.las'):
        inputs.append(key['Key'].split('/')[1])

print(inputs)
```

```
['out1.las', 'out10.las', 'out11.las', 'out12.las', 'out13.las', 'out14.las', 'out15.la
s', 'out16.las', 'out17.las', 'out18.las', 'out19.las', 'out2.las', 'out20.las', 'out21.
las', 'out3.las', 'out4.las', 'out5.las', 'out6.las', 'out7.las', 'out8.las', 'out9.la
s']
```

In [20]:

```
pipeline = """
[
  %s,
  {
    "type": "writers.tiledb",
    "array_name": "%s",
    "chunk_size": 1000000,
    "x_domain_st":295000.0,
    "y_domain_st":545000.0,
    "z_domain_st":-300.0,
    "x_domain_end":610000.0,
    "y_domain_end":925000.0,
    "z_domain_end": 5000.0,
    "append": %s
  }
]
"""

config = get_aws_config()
```

In [21]:

```
def append_array(inputs, s3_bucket, uri, aws_region, aws_key_id, aws_secret, create_sch
    import boto3
    import json
    import os
    import tempfile
    import tiledb

    os.environ['AWS_ACCESS_KEY_ID'] = aws_key_id
    os.environ['AWS_SECRET_ACCESS_KEY'] = aws_secret
    os.environ['AWS_DEFAULT_REGION'] = aws_region

    s3 = boto3.client('s3')
    for i in inputs:
        s3.download_file('lidar-backend', f"ingest/{i}", f"./{i}")

    task =  json.loads(pipeline % (','.join('"{0}"'.format(f) for f in inputs), uri, st
    print(task)
    with open('pipeline.json', 'w') as json_file:
      json.dump(task, json_file)

    os.system("pdal pipeline -i pipeline.json")

    return 1
```

In [22]:

```
start_node = Delayed(append_array, name='start_node', namespace='norman', image_name='3

# group remaining inputs
n = 2
tasks = [inputs[i:i+n] for i in range(1, len(inputs), n)]
nodes = []

config = get_aws_config()

for t in tasks:
    node = Delayed(append_array, namespace='norman', image_name='3.7-geo')(t, input_buc
    node.depends_on(start_node)
    nodes.append(node)
```

```
total_points = Delayed(np.sum, nodes, name='sum', namespace='norman')

start_node.visualize(force_plotly=False)
```

In [23]:

```
%%time
total = total_points.compute()
print(f"Processed {total * n} input las files")
```

```
Processed 20 input las files
CPU times: user 386 ms, sys: 53.5 ms, total: 439 ms
Wall time: 1min 18s
```

In [24]:

```
config = tiledb.Config()
config['sm.consolidation.mode'] = 'fragment_meta'
tiledb.consolidate(config=config, uri='s3://lidar-backend/nj_ingest')
print('done')
```

```
done
```

In [25]:

```
with tiledb.open(array_uri) as arr:
    print(arr.schema)
```

```
ArraySchema(
  domain=Domain(*[
    Dim(name='X', domain=(295000.0, 610000.0), tile='None', dtype='float64'),
    Dim(name='Y', domain=(545000.0, 925000.0), tile='None', dtype='float64'),
    Dim(name='Z', domain=(-300.0, 5000.0), tile='None', dtype='float64'),
  ]),
  attrs=[
    Attr(name='Intensity', dtype='uint16', var=False, nullable=False, filters=FilterList
([Bzip2Filter(level=5), ])),
    Attr(name='ReturnNumber', dtype='uint8', var=False, nullable=False, filters=FilterLi
st([ZstdFilter(level=7), ])),
    Attr(name='NumberOfReturns', dtype='uint8', var=False, nullable=False, filters=Filte
rList([ZstdFilter(level=7), ])),
    Attr(name='ScanDirectionFlag', dtype='uint8', var=False, nullable=False, filters=Fil
terList([Bzip2Filter(level=5), ])),
    Attr(name='EdgeOfFlightLine', dtype='uint8', var=False, nullable=False, filters=Filt
erList([Bzip2Filter(level=5), ])),
    Attr(name='Classification', dtype='uint8', var=False, nullable=False, filters=Filter
List([GzipFilter(level=9), ])),
    Attr(name='ScanAngleRank', dtype='float32', var=False, nullable=False, filters=Filte
rList([Bzip2Filter(level=5), ])),
    Attr(name='UserData', dtype='uint8', var=False, nullable=False, filters=FilterList
([GzipFilter(level=9), ])),
    Attr(name='PointSourceId', dtype='uint16', var=False, nullable=False, filters=Filter
List([Bzip2Filter(level=-1), ])),
    Attr(name='GpsTime', dtype='float64', var=False, nullable=False, filters=FilterList
([ZstdFilter(level=7), ])),
    Attr(name='Red', dtype='uint16', var=False, nullable=False, filters=FilterList([Zstd
Filter(level=7), ])),
    Attr(name='Green', dtype='uint16', var=False, nullable=False, filters=FilterList([Zs
tdFilter(level=7), ])),
    Attr(name='Blue', dtype='uint16', var=False, nullable=False, filters=FilterList([Zst
dFilter(level=7), ])),
  ],
  cell_order='hilbert',
  tile_order='NA',
  capacity=100000,
  sparse=True,
  allows_duplicates=True,
  coords_filters=FilterList([ZstdFilter(level=7)]),
)
```

In [ ]: