Redshift Job Submission

The Maya BatchRender Redshift submission UI utilizes the redshiftGPUCmdrange jobtype. The key feature of the redshiftGPUCmdrange jobtype is that it allows the user to specify the number of GPUs to use per instance, and leaves Qube to dynamcially allocate GPU affinity.

For example, a single worker with 4 GPUs could run a 1 GPU render and a 3 GPU render simultaneously, and upon completion of the 3 GPU render pick up and start rendering an additional 1 GPU render and 2 GPU render.

Each time the jobtype is run it makes a list of the available GPUs and randomly selects the ones it will use, this should hopefully ensure each GPU gets an even amount of usage over time.

In order to utilize the redshiftGPUCmdrange jobtype you need to make a few additions to your worker configurations. In each GPU enabled worker's qb.conf or in the central qbwrk.conf config you need to add a GPU_lock property, and a numbered GPU resource for each active GPU.
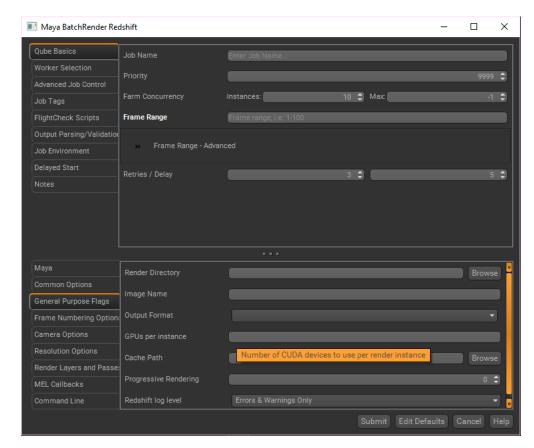
4 GPU Workers

```
[render0001]
worker_properties = host.GPU_lock=
worker_resources = host.GPU_0=1,host.GPU_1=1,host.GPU_2=1,host.GPU_3=1
[render0002]
worker_properties = host.GPU_lock=
worker_resources = host.GPU_0=1,host.GPU_1=1,host.GPU_2=1,host.GPU_3=1
```

2 GPU Workers

```
[render0003]
worker_properties = host.GPU_lock=
worker_resources = host.GPU_0=1,host.GPU_1=1
[render0004]
worker_properties = host.GPU_lock=
worker_resources = host.GPU_0=1,host.GPU_1=1
```

Then when submitting a redshift render you can specify the number of **GPUs per instance**:

The default value is 1 GPU per instance.

There is a caveat of the new jobtype, in the process of acquiring GPUs we currently use a 120 second time out to guarantee the instance has taken control of the GPU_lock. Acquiring the lock happens once per instance at the start of a render, and while we should be able to reduce this time in future release, for now it is a small time penalty that we must pay.