Using Flux to schedule and run jobs

To use flux, you first need to get an LSF allocation. Within the allocation, you will launch flux as if you launch your MPI job using jsrun or lrun. Upon successful launching of flux, you can submit your jobs into this flux instance. The following shows an example using an interactive allocation:

1. Get a 4-node LSF allocation

```
% bsub -Is -nnodes 4 -XF -G guests /usr/bin/tcsh
```

2. Launch your flux instance in the new shell prompt

```
% module load hwloc/1.11.10-cuda
% jsrun -a 1 -c ALL_CPUS -g ALL_GPUS --bind=none -n 4 /usr/bin/flux start
ip.sh
% # if run is preferred
% lrun -T 4 -N 4 --nolbind /usr/bin/flux start ip.sh
```

- module load hwloc/1.11.10-cuda will ensure the correctly configured libhwloc.so will be use for flux
- /usr/global/tools/pmi4pmix/blueos_3_ppc64le_ib/lib/libpmi.so is a library that allows flux to talk to jsrun's PMIX to bootstrap itself
- ip.sh is the initial program that gets executed when flux is launched across those 4 allocated node. It will write the hostname and URI pair for you to connect to submit jobs and use other flux commands:

```
% cat ip.sh
    #!/bin/sh
    echo "ssh://$(hostname)$(flux getattr broker.rundir)" | tee flux.info
    sleep inf
```

If you would like to change the location of the persistence directory (where the KVS-backing content store is located), you can start Flux with the `--setattr=persist-directory` argument:

```
% jsrun -a 1 -c ALL_CPUS -g ALL_GPUS --bind=none -n 4 /usr/bin/flux start -
o,--setattr=persist-directory=$PERSISTDIR
```

Relevant Flux documentation: persist-directory attribute and example usage of setting persist-directory when starting Flux

3. Connect to your flux instance from within a login node to use its services

- First look at the output of ip.sh to get the URI information (or cat flux.info)

```
% jsrun -a 1 -c ALL_CPUS -g ALL_GPUS --bind=none -n 4 /usr/bin/flux start
ip.sh
  ssh://sierra1211/var/tmp/flux-hb7VrK
```

- From within a login node (e.g., sierra4362), use flux-proxy command's ssh connector to connect to this URI:

```
% flux proxy ssh://sierra1211/var/tmp/flux-hb7VrK
% # if you don't want to copy-and-paste
% flux proxy `cat flux.info`
```

- At a new shell created, issue flux commands such as **flux-submit** to submit your jobs, **flux-wreck-ls** to list the jobs and **flux-wreck-attach** to print out the job's stdout and stderr. For example, to submit two hostname jobs at 4 nodes 4 tasks and print the output of this job:

```
% flux submit -N 4 -n 4 /usr/bin/hostname
submit: Submitted jobid 1
% flux submit -N 4 -n 4 /usr/bin/hostname
submit: Submitted jobid 2

% flux wreck attach 1
sierra3238
sierra3741
sierra3742
sierra3237


% flux wreck attach 2
sierra3238
sierra3741
sierra3742
sierra3237
```

- To query the status of these jobs:

```
% sierra4358{dahn}26: flux wreck ls
ID NTASKS STATE  START               RUNTIME RANKS COMMAND
 1 4      exited 2018-07-03T20:10:28 0.241s  [0-3] hostname
 2 4      exited 2018-07-03T20:13:31 0.101s  [0-3] hostname
```

- To submit two GPU jobs each with 4 tasks per node across 4 nodes, each uses 10 cores and 1 GPU:

```
% flux submit -N 4 -n 4 -c 10 -g 1 /usr/bin/hostname
submit: Submitted jobid 3
% flux submit -N 4 -n 4 -c 10 -g 1 /usr/bin/hostname
submit: Submitted jobid 4
```

- To launch an MPI application, users have to specify an MPI "personality" to work around some interoperability issues between Flux and Spectrum MPI:

```
% flux submit -N 4 -o mpi=spectrum g/g0/dahn/testcases/parallel_dbg_target
/parallel_dbg_target/virtual_ring_mpi
```

If you are submitting jobs via a flux proxy, the above command will work correctly.
If you are submitting jobs via a script that is running within the flux instance (e.g., ip.sh), you will first need to unset variables set by jsrun (i.e., anything that starts with OMPI_, JSM_, or PMIX_).

For more information, please see the following issue on GitHub : https://github.com/flux-framework/flux-core/issues/1619

- Other usage examples that help facilitate an easy workflow management can be found on here.