Running Flux in Slurm

Flux development team: Al Chu, James Corbett, Jim Garlick, Mark Grondona, Dan Milroy, Chris Moussa, Tom Scogland, Vanessa Sochat, Jae-Seung Yeom

Sept 13, 2023

Ryan Day
LC Resource Management
What is Flux?

- Flux is a next generation open source resource manager being developed at LLNL.
- Flux is hierarchical. Every flux batch job is a full flux instance with the ability to schedule more jobs on its resources.
- Flux python API gives users powerful tools for running complex workflows.
- Flux can be easily run by users in their Slurm allocations.
What is Flux?

- Flux is a next generation open source resource manager being developed at LLNL.*

- Flux is hierarchical. Every flux batch job is a full flux instance with the ability to schedule more jobs on its resources.

- Flux python API gives users powerful tools for running complex workflows.

- Flux can be easily run by users in their Slurm allocations.

*Flux is NOT a replacement for Slurm.**

**yet**
Flux on Github

https://github.com/flux-framework
What is Flux?

- Flux is a next generation open source resource manager being developed at LLNL.

- Flux is hierarchical. Every flux batch job is a full flux instance with the ability to schedule more jobs on its resources.

- Flux python API gives users powerful tools for running complex workflows.

- Flux can be easily run by users in their Slurm allocations.
Flux is hierarchical: Launching steps in Slurm

sbatch

Allocation (4 nodes)

srun

Job step (2 nodes)

Job step (2 nodes)

srun

srun

Job step (1 node)

Job step (1 node)
Flux is hierarchical: Launching jobs in Flux

- Flux batch
  - Job (4 nodes)
    - Flux batch
      - Job (2 nodes)
        - Flux run
          - Job (1 node)
        - Flux run
          - Job (1 node)
    - Flux batch
      - Job (2 nodes)
        - Flux run
          - Job (1 node)
Digression: a personal timeline of simulation

2000: 1 simulation takes many jobs.
2002: still waiting for those jobs.
2004: run 100 copies of 1 simulation across many jobs.
2008: build statistical models instead of running simulations.
2010: have kids and go work in HPC support
Digression: a personal timeline of simulation

2000: 1 simulation takes many jobs.

2002: still waiting for those jobs.

2004: run 100 copies of 1 simulation across many jobs.


2008: build statistical models instead of running simulations.

2010: have kids and go work in HPC support
Digression: a personal timeline of simulation

2000: 1 simulation takes many jobs.
2002: still waiting for those jobs.
2004: run 100 copies of 1 simulation across many jobs.
2008: build statistical models instead of running simulations.
2010: have kids and go work in HPC support
Digression: a personal timeline of simulation

- 2000: 1 simulation takes many jobs.
- 2004: run 100 copies of 1 simulation across many jobs.
- 2008: build statistical models instead of running simulations.
- 2010: have kids and go work in HPC support
Digression: a personal timeline of simulation

- 2000: 1 simulation takes many jobs.
- 2004: run 100 copies of 1 simulation across many jobs.
- 2008: build statistical models instead of running simulations.
- 2010: have kids and go work in HPC support
Digression: a personal timeline of simulation

2000: 1 simulation takes many jobs.
2002: still waiting for those jobs.
2004: run 100 copies of 1 simulation across many jobs.
2008: build statistical models instead of running simulations.
2010: have kids and go work in HPC support

2018
MuMMI implements a complex and dynamic workflow

**Single Macro Simulation**
- C++ (with MPI); MOOSE; ddcMD
- 24 CPU cores/node; 2400 MPI tasks
- 242 GB per day

**ML-based Selection**
- Python; ML frameworks; FAISS
- 24 CPU cores
- >1000 decisions per minute

**CG Setup**
- Python; Custom; GROMACS
- 24 CPU cores each
- 1.5 hr each

**CG Simulation**
- C++ (with CUDA); ddcMD
- 1 GPU + 1 CPU core each
- 1.04 µs and ~6.5 GB per day

**Analysis Aggreg. & Feedback**
- Python; Custom
- 24 CPU cores
- >2K frames per day

**In situ CG Analysis**
- Python; Custom
- 3 CPU cores each
- 120,000 reads per cycle

Flux is hierarchical: Sierra node diagram

Sierra Node

gpu
gpu

memory

20 cores

memory

20 cores

Local storage

network
Flux is hierarchical: ATS node diagram

Flux instance

Sierra Node

Flux instance

Flux instance

Local storage

memory

2 cores

18 cores

18 cores

memory

GPU

GPU

50 GB/s per channel (bidirectional)

150 GB/s aggregated bandwidth (bidirectional)

128 GB

170 GB/s aggregate peak bandwidth

PCIe Gen4 x4

PCIe Gen4 x4

PCIe Gen2 x4

PCIe Gen2 x4

PCIe Gen2 x1

PCIe Gen4 x8 CAPI

PCIe Gen4 x8 CAPI

PCIe Gen4 x16 - CAPI

X Bus

64 GB/s

Network

Lawrence Livermore National Laboratory

LLNL-PRES-853734
Flux is hierarchical: other workflow examples

- Uncertainty quantification (UQ): run an ensemble of simulation pipelines with different starting conditions.

- Automated testing / CI: running arbitrary sets of unit tests for parallel programs.
What is Flux?

- Flux is a next generation open source resource manager being developed at LLNL.

- Flux is hierarchical. Every flux batch job is a full flux instance with the ability to schedule more jobs on its resources.

- Flux python API gives users powerful tools for running complex workflows.

- Flux can be easily run by users in their Slurm allocations.
Usability: Submitting a job

- **Slurm**
  
  `srun -N2 -n4 -t 2:00 sleep 120`

- **Flux CLI**
  
  `flux run -N2 -n4 -t 2m sleep 120`

- **Flux API:**
  
  ```python
  import flux
  from flux.job import JobspecV1

  f = flux.Flux()
  j = JobspecV1.from_command(command=['sleep', '120'],
                              num_nodes=2,
                              num_tasks=4)
  j.set_duration(120)
  resp = flux.job.submit(f, j)
  ```

from flux.job import JobspecV1, FluxExecutor

def event_callback(future, event):
    print(f"job {future.jobid()} triggered event {event.name!r}")

def main():
    # set up jobspecs
    ...
    # submit jobs and register event callbacks for all events
    with FluxExecutor() as executor:
        futures = [executor.submit(compute_jobreq) for _ in range(args.njobs // 2)]
        ...
        for fut in futures:
            # each event can have a different callback
            for event in executor.EVENTS:
                fut.add_event_callback(event, event_callback)

What is Flux?

- Flux is a next generation open source resource manager being developed at LLNL.
- Flux is hierarchical. Every flux batch job is a full flux instance with the ability to schedule more jobs on its resources.
- Flux python API gives users powerful tools for running complex workflows.
- Flux can be easily run by users in their Slurm allocations.
Running Flux: starting Flux in a Slurm allocation

You can start flux in a Slurm allocation on any cluster:

```
[day36@quartz2306:~]$ salloc -N2 -p pdebug
salloc: Pending job allocation 1228472
salloc: job 1228472 queued and waiting for resources
salloc: job 1228472 has been allocated resources
salloc: Granted job allocation 1228472
salloc: Waiting for resource configuration
salloc: Nodes quartz[44-45] are ready for job

sh-4.4$ srun -N2 -n2 --pty flux start

sh-4.4$ flux resource list

<table>
<thead>
<tr>
<th>STATE</th>
<th>NNODES</th>
<th>NCORES</th>
<th>NGPUS</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>2</td>
<td>72</td>
<td>0</td>
<td>quartz[44-45]</td>
</tr>
<tr>
<td>allocated</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>down</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

sh-4.4$ flux run -N2 -n2 hostname
quartz44
quartz45

sh-4.4$
```

Running Flux: running a batch script

```bash
sh-4.4$ cat quickexample.sh
#!/bin/sh
# flux: -N 2
# flux: -n 2
date
flux run -n 2 ~/hello/mpi_hello/hello_mpi

sh-4.4$ srun -N2 -n2 flux start \\ 'flux batch quickexample.sh && flux queue drain'
f2SqkPL3
sh-4.4$ cat flux-f2SqkPL3.out
Fri Aug 25 13:37:05 PDT 2023
Hello from task 1 on quartz45!
Hello from task 0 on quartz45!

Number of MPI tasks is: 2

sh-4.4$
```
Running Flux: running a python workflow

```bash
sh-4.4$ cat flux-workflow-examples/job-status-control/bookkeeper.py
#!/usr/bin/env python3
...
  def event_callback(future, event):
    print(f"job {future.jobid()} triggered event {event.name!r}")
...
sh-4.4$ srun -N2 -n2 flux start ./flux-workflow-examples/job-status-control/bookkeeper.py 2
bookkeeper: all jobs submitted
bookkeeper: waiting until all jobs complete
job f2RSD4q5 triggered event 'submit'
job f2RSD4q6 triggered event 'submit'
job f2RSD4q5 triggered event 'depend'
job f2RSD4q5 triggered event 'priority'
job f2RSD4q5 triggered event 'exception'
job f2RSD4q5 triggered event 'clean'
job f2RSD4q6 triggered event 'depend'
job f2RSD4q6 triggered event 'priority'
job f2RSD4q6 triggered event 'exception'
job f2RSD4q6 triggered event 'clean'
bookkeeper: all jobs completed
```

Running Flux: getting Flux

- Install with spack:
  `spack install flux-sched`

- Build from source:
  `git clone https://github.com/flux-framework/flux-core.git`
  configure, make, make install

  `git clone https://github.com/flux-framework/flux-sched.git`
  configure, make, make install

- Docker (quick single node deployment):
  `docker run -ti fluxrm/flux-sched:latest`

Where to find out more

- https://github.com/flux-framework/
- https://github.com/flux-framework/Tutorials
- https://hpc-tutorials.llnl.gov/flux/
- `flux help`, `flux help run`, `flux help jobs`, ...