Introducing Flux

A scalable resource manager for LC clusters

Ryan Day
LC Operational Resource Management

Flux development team: Dong Ahn, Stephen Herbein, Jim Garlick, Mark Grondona, Al Chu, Chris Moussa, Dan Milroy

December 8, 2020
What is flux?

- Flux is a modular, fully hierarchical resource manager and job scheduler.
- Modular development model allows a rich and consistent API which makes it easy to launch flux instances from within scripts.
- Fully hierarchical means that every flux ‘job step’ can be a full flux instance with the ability to schedule more job steps on its resources.
- Flux can be used now on LC systems.

Flux uses a new model for scheduling
What is flux?

- Flux is a modular, fully hierarchical resource manager and job scheduler.
- Modular development model allows a rich and consistent API which makes it easy to launch flux instances from within scripts.
- Every flux ‘job step’ can be a full flux instance with the ability to schedule more job steps on its resources.
- Flux can be used now on LC systems.

Flux has a rich API
Usability: Submitting a Batch Job

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

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

- **Flux API**

```python
import json, flux, job
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)
```

Scalability: Running Many Jobs

- **Slurm / CLI**
  - `find ./ -exec sbatch -N1 tar -cf {{}.tgz} {}\;`
    - Slow: requires acquiring a lock in Slurm, can timeout causing failures
    - Inefficient: uses 1 node for each task
  - `find ./ -exec srun -n1 tar -cf {{}.tgz} {}\;`
    - Slow: spawns a process for every submission
    - Inefficient: is not a true scheduler

- **Flux API**

```python
flux start my_jobs.py

import flux, flux.job
from flux.job import JobspecV1

h = flux.Flux()
for f in os.listdir('.'):
    command = ['tar', '-cf', '{}.tgz'.format(f), f]
    flux.job.submit(h, JobspecV1.from_command(command))
```

What is flux?

- Flux is a modular, fully hierarchical resource manager and job scheduler.
- Modular development model allows a rich and consistent API which makes it easy to launch flux instances from within scripts.
- Every flux ‘job step’ can be a full flux instance with the ability to schedule more job steps on its resources.
- Flux can be used now on LC systems.
Flux is hierarchical: Launching steps in Slurm

Complex schedulers allow complex workflows
Flux is hierarchical: Launching instances in Flux

- Get allocation (launch instance)
- Launch instance
  - Instance (4 nodes)
- Launch instance
  - Instance (2 nodes)
- Launch instance
  - Instance (2 nodes)
- Launch instance
  - Instance (2 node)
- Launch instance
  - Instance (2 node)

Complex schedulers allow complex workflows
Scalability: Running Millions of Jobs

- **Single Flux Instance**
  - `flux start my_workflow.py`

- **Statically Partitioned Flux Instances**
  - `for x in $(seq 1 $num_nodes); do
    flux mini batch -N1 flux start my_workflow_$x.py
  done`

- **Flux Hierarchy**
  - `flux-tree -N $num_nodes \`
    -T ${num_nodes} \`
    -J $num_jobs -- jobspec.yaml`
  - `flux-tree -N $num_nodes \`
    -T ${num_nodes}x${cores_per_node} \`
    -J $num_jobs -- jobspec.yaml`

Flux is hierarchical: CTS node diagram

Traditional scheduling maps well to simple nodes
Flux is hierarchical: ATS node diagram

New systems are more complex and heterogeneous
Flux is hierarchical: ATS node diagram

~10% of cycles on node
Flux is hierarchical: ATS node diagram

~90% of cycles on node
Flux is hierarchical: ATS node diagram

~100% of cycles on node
MuMMI implements a complex workflow to enable a new genre of multiscale simulation for cancer research

**Multiscale Machine-Learned Modeling Infrastructure (MuMMI)**
- Novel framework coupling multiple scales using a hypothesis driven selection process.

MuMMLI 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
120,000 reads per cycle

FIFO; real-time tracking & update

In situ CG Analysis
Python; Custom
3 CPU cores each
>2K frames per day

The high through-put, low latency scheduling enables fast restarts and consistent utilization of all resources

- Hierarchical scheduling allows MuMMI to reach steady state in ~45 minutes (newer versions will reduce turnaround time)

- Depending on the scientific hypothesis MuMMI utilizes >95% of the available compute

What is flux?

- Flux is a modular, fully hierarchical resource manager and job scheduler.

- Modular development model allows a rich and consistent API which makes it easy to launch flux instances from within scripts.

- Every flux ‘job step’ can be a full flux instance with the ability to schedule more job steps on its resources.

- Flux can be used now on LC systems.
Using Flux: getting flux

- Installed on LC systems
  
  `[day36@rzslc4:~]$ ls /usr/global/tools/flux/$SYS_TYPE/default/bin
  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

Using Flux: starting an instance

[day36@rzalastor2:~]$ salloc -N4 --exclusive
salloc: Granted job allocation 220682
sh-4.2$ flux keygen
Saving /g/g0/day36/.flux/curve/client
Saving /g/g0/day36/.flux/curve/client_private
Saving /g/g0/day36/.flux/curve/server
Saving /g/g0/day36/.flux/curve/server_private
sh-4.2$ srun -N4 -n4 --pty flux start
sh-4.2$ flux mini run -n4 hostname
rzalastor16
rzalastor15
rzalastor17
rzalastor14
sh-4.2$

Using Flux: running a batch script

sh-4.2$ cat quickexample.sh
#!/bin/sh
flux mini batch -N 2 -n 2 --wrap << EOF
date
flux mini run -n 2 ~/hello/hello_mpi
EOF
sh-4.2$ ./quickexample.sh
f4aDXvqSo
sh-4.2$ flux jobs -f completed,failed
  JOBID  USER   NAME       ST  NTASKS NNODES  RUNTIME  RANKS
  f4aDXvqSo day36  batchscrip CD  2  2  4.302s [0-1]
  f5jrprGw  day36   hostname CD  4  4  0.098s [0-3]
sh-4.2$ cat flux-f4aDXvqSo.out
Tue Dec  1 12:05:21 PST 2020
Hello from task 0 on rzalastor14!
MASTER: Number of MPI tasks is: 2
Hello from task 1 on rzalastor15!
sh-4.2$

Where to find out more

CLI

- Man flux-mini, man flux-jobs, etc.

API / Workflow

- [https://github.com/LLNL/maestrowf](https://github.com/LLNL/maestrowf) — [https://lc.llnl.gov/confluence/display/MAESTRO/Maestro+Home](https://lc.llnl.gov/confluence/display/MAESTRO/Maestro+Home)
- Email [lc-hotline@llnl.gov](mailto:lc-hotline@llnl.gov) with questions, bugs, or to get in touch with the workflows team.

Questions?
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.