Flux: A Next-Generation Resource Manager for HPC and Beyond

NMEWS3 at IPAM

James Corbett

LLNL-PRES-848710

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

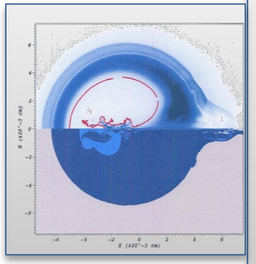
Flux is a resource manager and job scheduler developed at LLNL

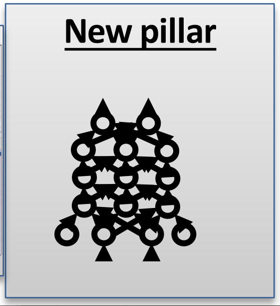
- Flux is a HPC resource manager like Slurm, LSF, or PBS Pro
- Used for requesting resource allocations and scheduling/launching jobs
- Currently used by many individual users and workflow systems at LLNL, LBNL, and ORNL

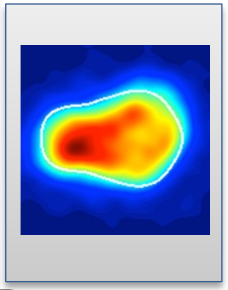
Flux is in the early stages of deployment at LLNL

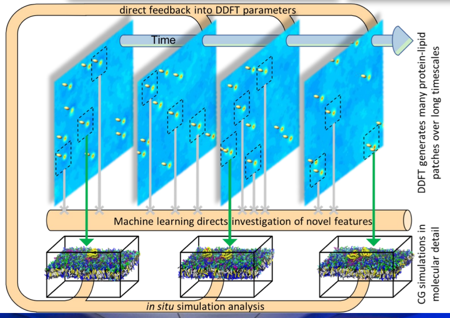
- Flux is currently deployed on a number of LLNL clusters
  - Tioga, Hetchy, Corona, Fluke, Elmerfudd, and, just this week, RZVernal and Tenaya
  - Generally smaller testbed clusters, to gather user feedback (although three are in the top 200 on the Top500 list)
- Lots of ongoing work to prepare users for Flux on more and more LLNL clusters (El Cap in particular)

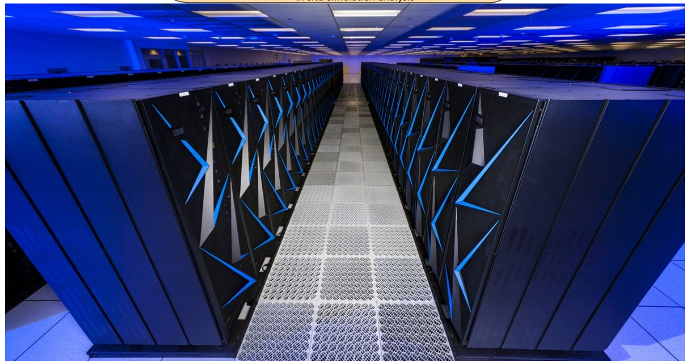
Why another resource manager?











- Co-scheduling
- Job throughput
- Job communication/coordination
- Portability
- Extremely heterogenous resources

Pre-exascale scientific workflows strain the capabilities of traditional HPC resource managers and schedulers

Co-scheduling:

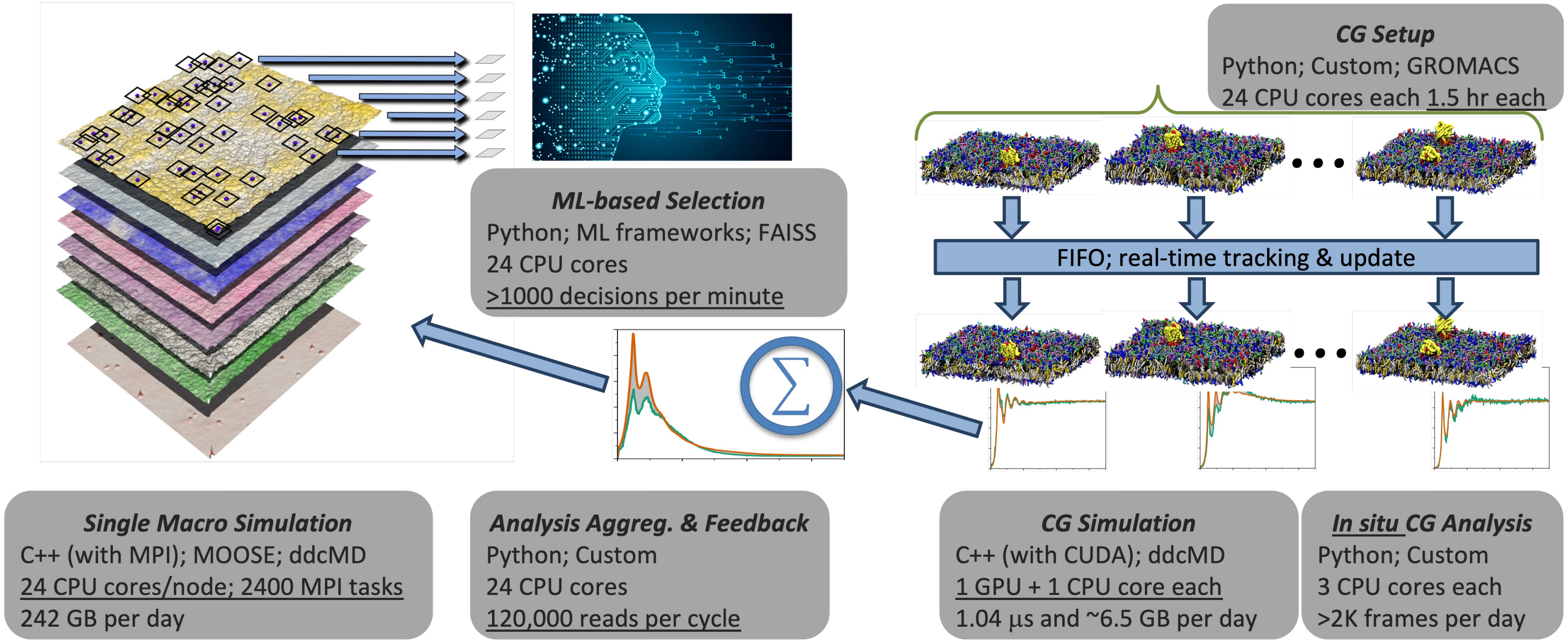
CG, analysis bound to cores nearest PCIe buses

Job comms/coordination:

36,000 concurrent tasks; 176,000 cores, 16,000 GPUs

Portability:

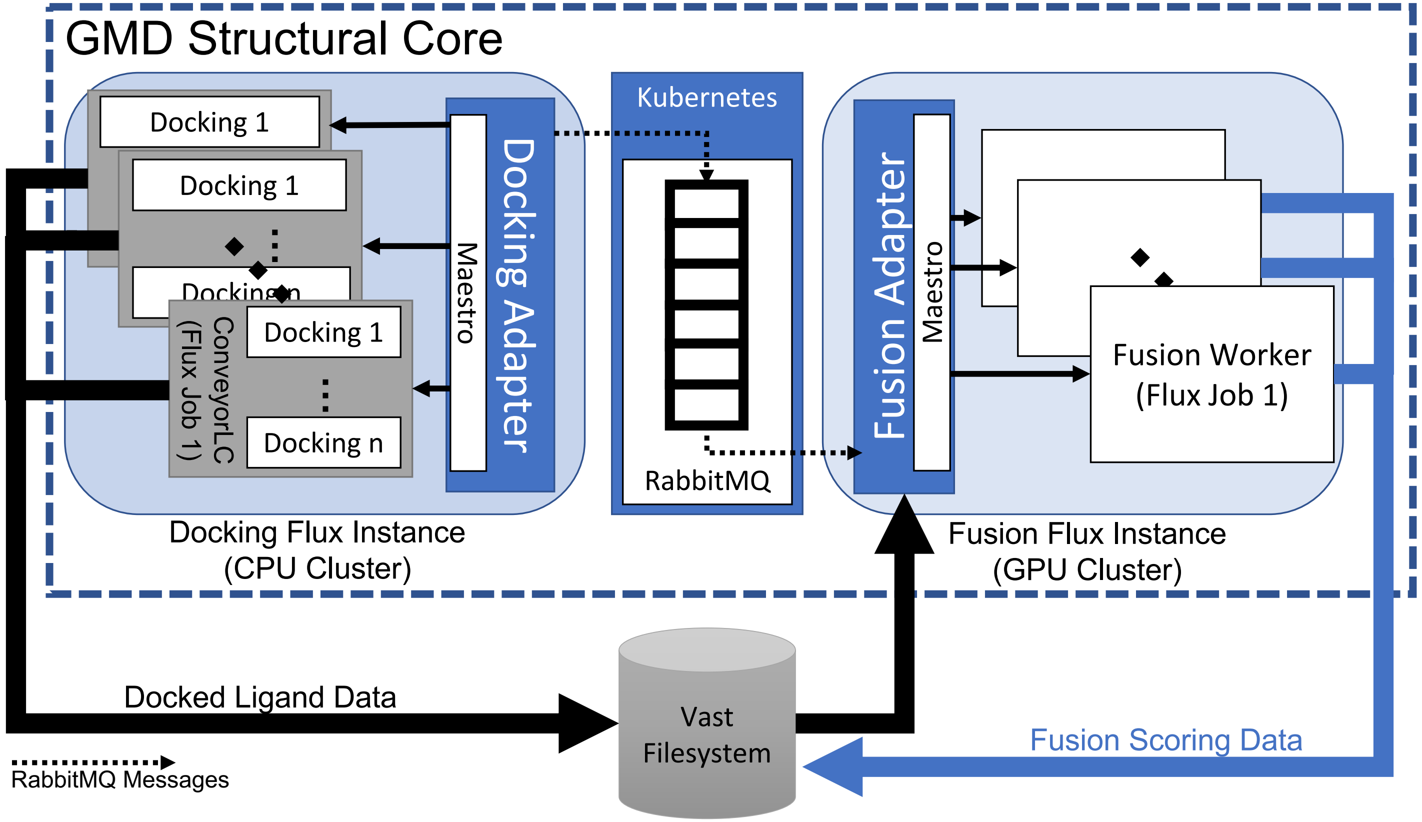
adapt tasks to different schedulers/managers



MuMMI: SC'19 best paper, SC'21 paper

MPI-based simulation with in-situ analysis plus ML

Next-generation, cross-cluster scientific workflows are demanding portability and cloud integration.



AHA MoleS: eScience'22 best paper

Complex workflows integrating cloud technologies at LLNL and beyond

- Scalable message broker couples MPI-based tasks, analysis, workflow runs anywhere (**AHA MoleS**)
- HPC simulation with AI/ML surrogates, orchestrated databases (AMS)
- Many other examples: ATOM, AMPL, GMD, etc.

2020 lab survey found that 73% of LLNL workflows interested in cloud integration

MPI-based simulation with analysis, ML, and containerized components

Flux offers a suite of features and behavior to allow it to adapt to future needs

- Relatively easy for a Slurm user to learn
- Sophisticated, configurable scheduling abilities
- Scalable performance
- Consistent behavior across centers
- Easy to build (few dependencies)
  - Conda, spack packages
  - Docker containers for experimentation
- Interfaces designed with workflow tools in mind
- The ability to nest Flux instances inside other Flux instances, or allocations from other resource managers
- Kubernetes integrations
- Well documented (???)

Flux's interface is often intentionally similar to Slurm's

```
#!/bin/bash
# flux: -N 4
# flux: -t 90
# flux: -q pdebug
# flux: --job-name=myjob

flux run -n16 hostname
```

```
#!/bin/bash
#SBATCH -N 4
#SBATCH -t 90
#SBATCH -p debug
#SBATCH -J my_job

srun -n16 hostname
```

- Submitting a batch script
  - flux batch script.sh
  - sbatch script.sh
- Getting an interactive allocation
  - flux alloc –N 4 –q debug –t 90
  - salloc –N 4 –p debug –t 90
- Launching an MPI job
  - flux run -N2 –n16 –c2 –g2
  - srun –N2 –n16 –c2 –gpus-per-task=2
- But there are plenty of differences

Command-Line Interface Cross-Reference

Command	LSF	Moab	Slurm	Flux*
submit a job	**bsub**	**msub**	**sbatch**	**flux batch**
submit an interactive job	**bsub -Is [bash|csh]**		**salloc**	**flux alloc**
submit an xterm job	**bsub -XF xterm**	**mxterm**	**sxterm**	**flux submit xterm**
launch parallel tasks	**mpirun/jsrun/lrun**		**srun**	**flux run / flux submit**
modify a pending job	**bmod *jobid***	**mjobctl -m *jobid***	**scontrol update job *jobid***	
hold a pending job	**bstop *jobid***	**mjobctl -h *jobid***	**scontrol hold *jobid***	**flux job urgency *jobid* hold**
release a held job	**bresume *jobid***	**mobctl -r *jobid***	**scontrol release *jobid***	**flux job urgency *jobid* default**
cancel a job	**bkill *jobid***	**canceljob *jobid***	**scancel *jobid***	**flux job cancel *jobid***
signal a job	**bkill -s *signal* *jobid***	**mobctl -N signal=*signal* *jobid***	**scancel -s *signal* *jobid***	**flux job kill -s *signal* *jobid***
show detailed job information	**bquery -l *jobid***	**checkjob *jobid***	**scontrol show job *jobid***	**flux job info *jobid***
show job queue	**bquery -u all**	**showq**	**squeue**	**flux jobs**
show historical jobs	**bhist**		**sacct**	**flux jobs -a**
show detailed historical job info	**bhist -l *jobid***		**sacct -l -j *jobid***	**flux job info *jobid***
show job priorities	**bquery -aps**	**mdiag -p**	**sprio**	**flux jobs -ao '{id} {priority}'**
show node resources	**bhosts**	**mdiag -n**	**scontrol show node**	**flux resource list / flux hwloc info**
show available queues	**bqueues**	**mdiag -c**	**sinfo**	**flux resource list**
show queue details	**bqueues -l**	**mdiag -c -v**	**scontrol show partition**	**flux queue list**
show charge accounts	**bugroup**	**mshare**	**sshare**	
show configuration settings	**bparams -a**	**mschedctl -l**	**scontrol show conf**	

A rich set of well-defined APIs enables easy job coordination and communication.

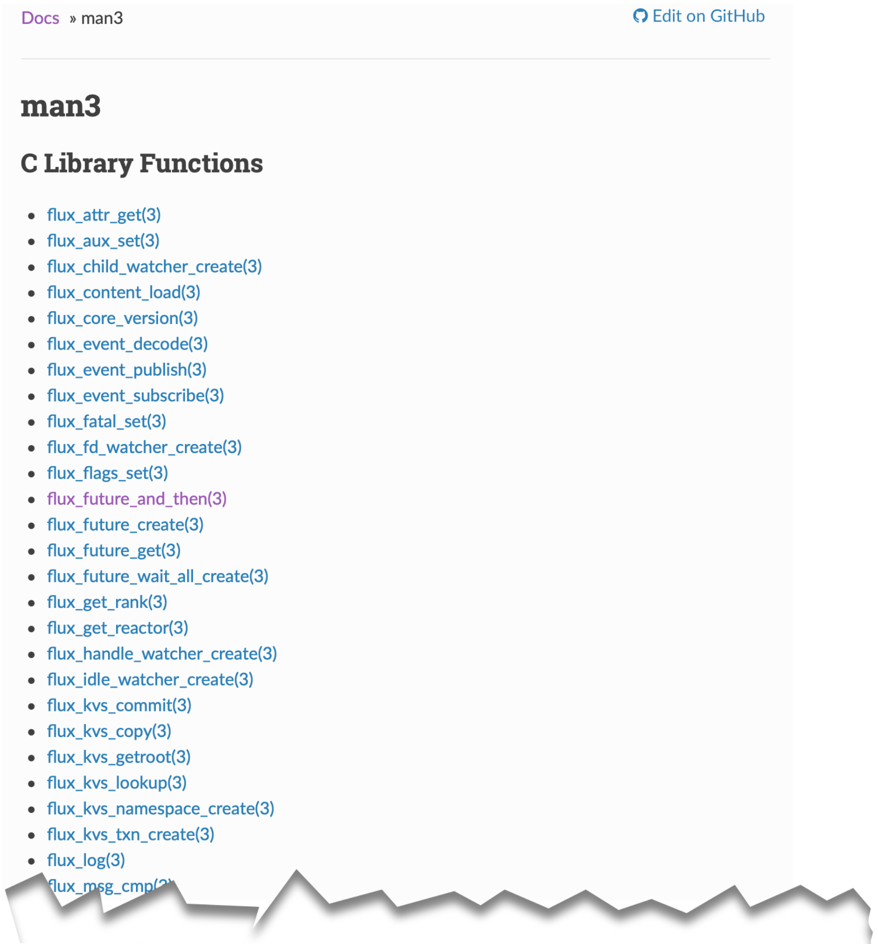
- Complete Python and C libraries

- Jobs in ensemble-based simulations often require close coordination and communication with the scheduler as well as among them.
  - Traditional CLI-based approach can be slow and cumbersome.
  - Ad hoc approaches (e.g., many empty files) can lead to many side effects.

- Flux provides well-known communication primitives.
  - Pub/sub, request-response, and send-recv patterns

- High-level services
  - Key-value store (KVS) API
  - Job API (submit, wait, state change notification, etc)

- Flux's APIs are consistent across different platforms


Docs » man3 Edit on GitHub

man3

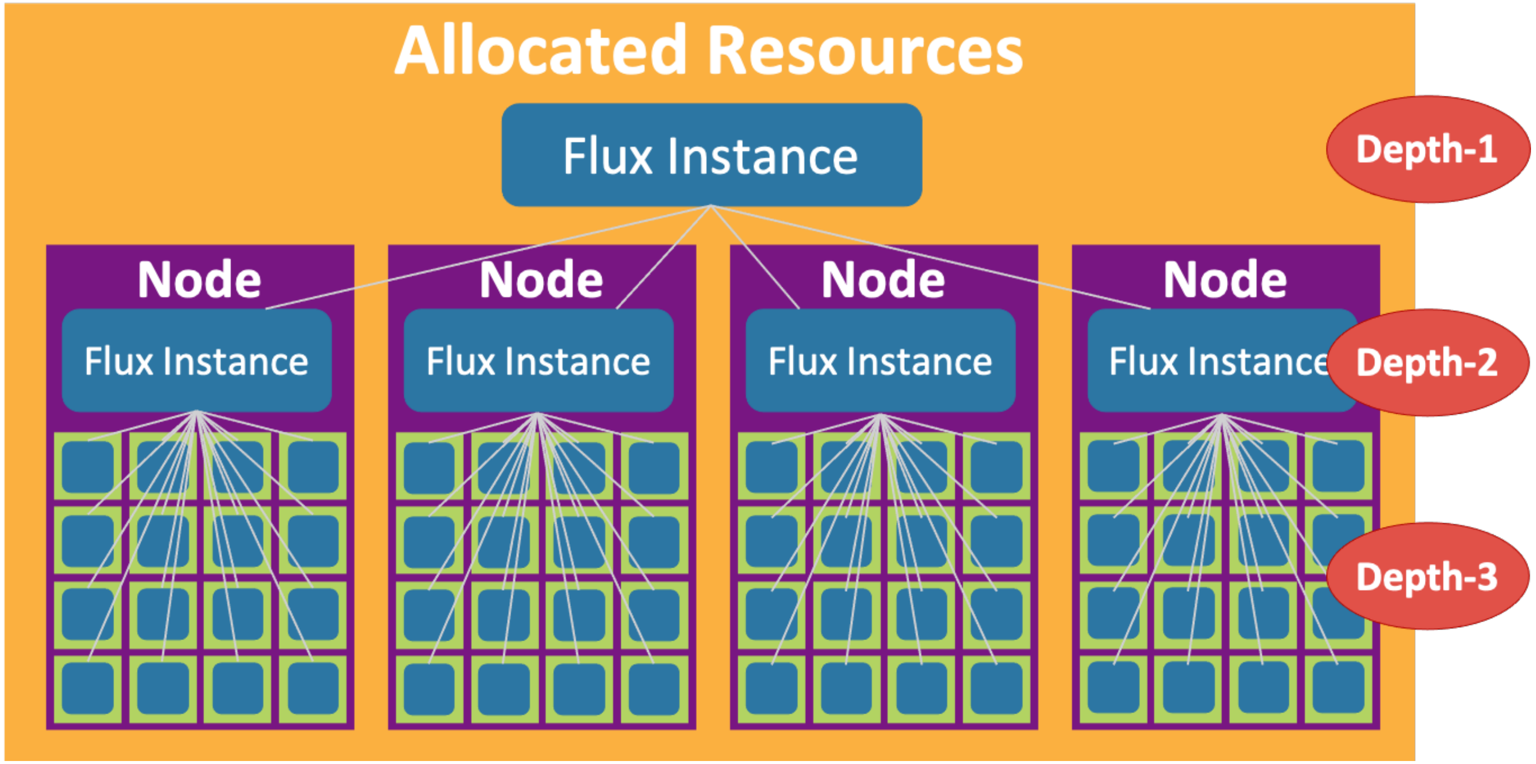
C Library Functions

- flux_attr_get(3)
- flux_aux_set(3)
- flux_child_watcher_create(3)
- flux_content_load(3)
- flux_core_version(3)
- flux_event_decode(3)
- flux_event_publish(3)
- flux_event_subscribe(3)
- flux_fatal_set(3)
- flux_fd_watcher_create(3)
- flux_flags_set(3)
- flux_future_and_then(3)
- flux_future_create(3)
- flux_future_get(3)
- flux_future_wait_all_create(3)
- flux_get_rank(3)
- flux_get_reactor(3)
- flux_handle_watcher_create(3)
- flux_idle_watcher_create(3)
- flux_kvs_commit(3)
- flux_kvs_copy(3)
- flux_kvs_getroot(3)
- flux_kvs_lookup(3)
- flux_kvs_namespace_create(3)
- flux_kvs_txn_create(3)
- flux_log(3)

Flux handles jobs differently from other resource managers

- Unlike many resource managers, when you launch a job, you get exactly the resources you ask for, and no more
  - Ask for four tasks and five cores per task, and your application will have four tasks, each bound to five cores
  - Unlike Slurm, which will (sometimes, depending on configuration/plugins) give you whole nodes
- All jobs are given exclusive sets of resources by default

Flux's fully-hierarchical approach enables scalable performance



- Flux can run inside of the allocations of other resource managers
- But it can also run inside of Flux allocations
- Full workflow-enablement support
  - Via hierarchical resource subdivision
  - Sub-resource manager per subdivision with service specialization
  - E.g., at LLNL: MuMMI, AHA MoleS, UQP

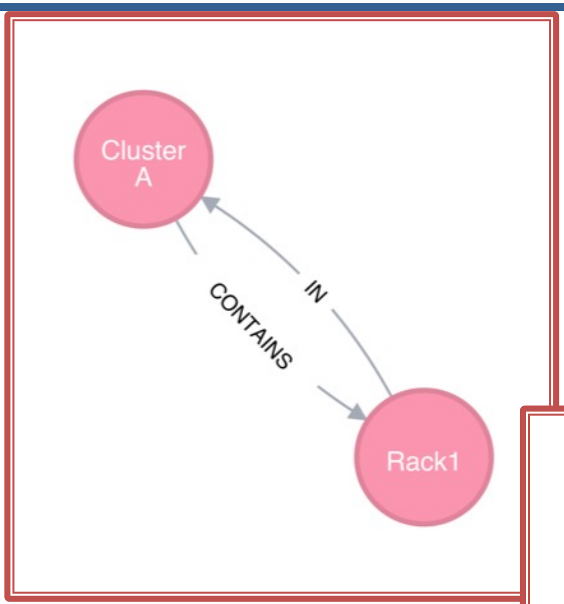
Flux pioneers directed graph-based scheduling to manage complex combinations of extremely heterogenous resources

- Traditional resource data models are largely ineffective for resource heterogeneity
  - designed when systems were simpler
  - node-centric models

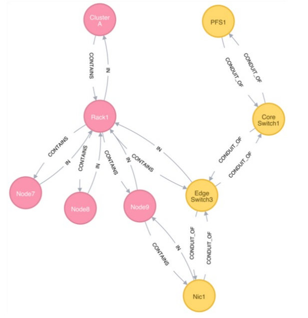
- Edges express relationships, flows

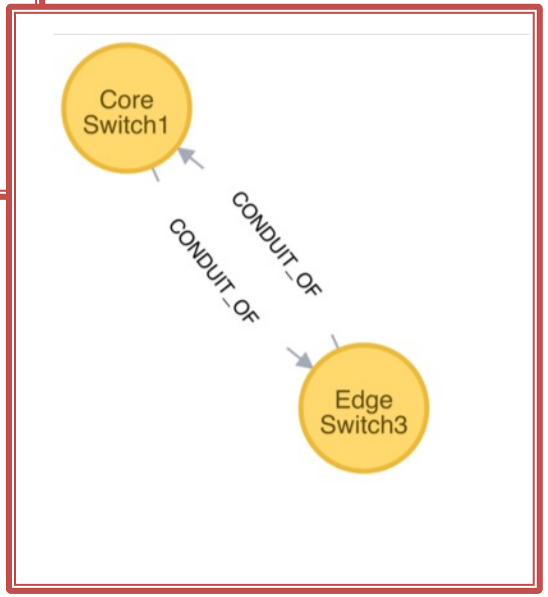
- Complex scheduling without changing scheduler code

- Rich, well-defined C, C++, Go (*in progress*) APIs



Containment subsystem





Network connectivity subsystem

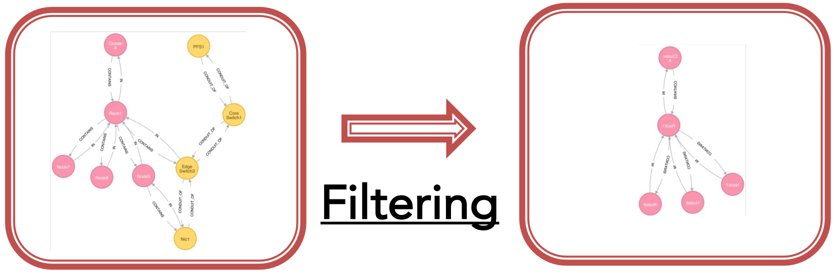
Flux uses graph filtering and pruned searching to manage the graph complexity and optimize our graph search

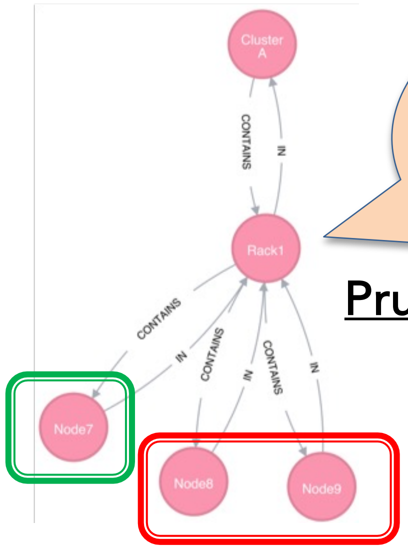
- The total graph can be quite complex
  - Two techniques to manage the graph complexity and scalability

1. Filtering reduces graph complexity
   - The graph model needs to support schedulers with different complexity
   - Provide a mechanism by which to filter the graph based on what subsystems to use

2. Pruned search increases scalability
   - Fast RB tree-based planner is used to implement a pruning filter per each vertex.
   - Pruning filter keeps track of summary information (e.g., aggregates) about subtree resources.
   - Scheduler-driven pruning filter update

Containment+Network   Containment





Pruning

Prune filter tracks available aggregate node count at the subtree

Flux's graph-oriented canonical jobspec allows for a highly expressive resource request specification

- Graph-oriented resource requirements
  - Express the resource requirements of a program to the scheduler
  - Express program attributes such as arguments, runtime, and task layout to the execution service
- cluster→ rack[2]→slot[3]→node[1]→socket[2]→core[18]
- ***slot*** is the only non-physical resource type
  - Represent a schedulable place where program processes will be spawned and contained
- Tasks section references slot and defines command

```yaml
version: 1
resources:
  - type: cluster
    count: 1
    with:
      - type: rack
        count: 2
        with:
          - type: slot
            label: myslot
            count: 3
            with:
              - type: node
                count: 1
                with:
                  - type: socket
                    count: 2
                    with:
                      - type: core
                        count: 18

# a comment
attributes:
  system:
    duration: 3600
tasks:
  - command: app
    slot: myslot
    count:
      per_slot: 1
```

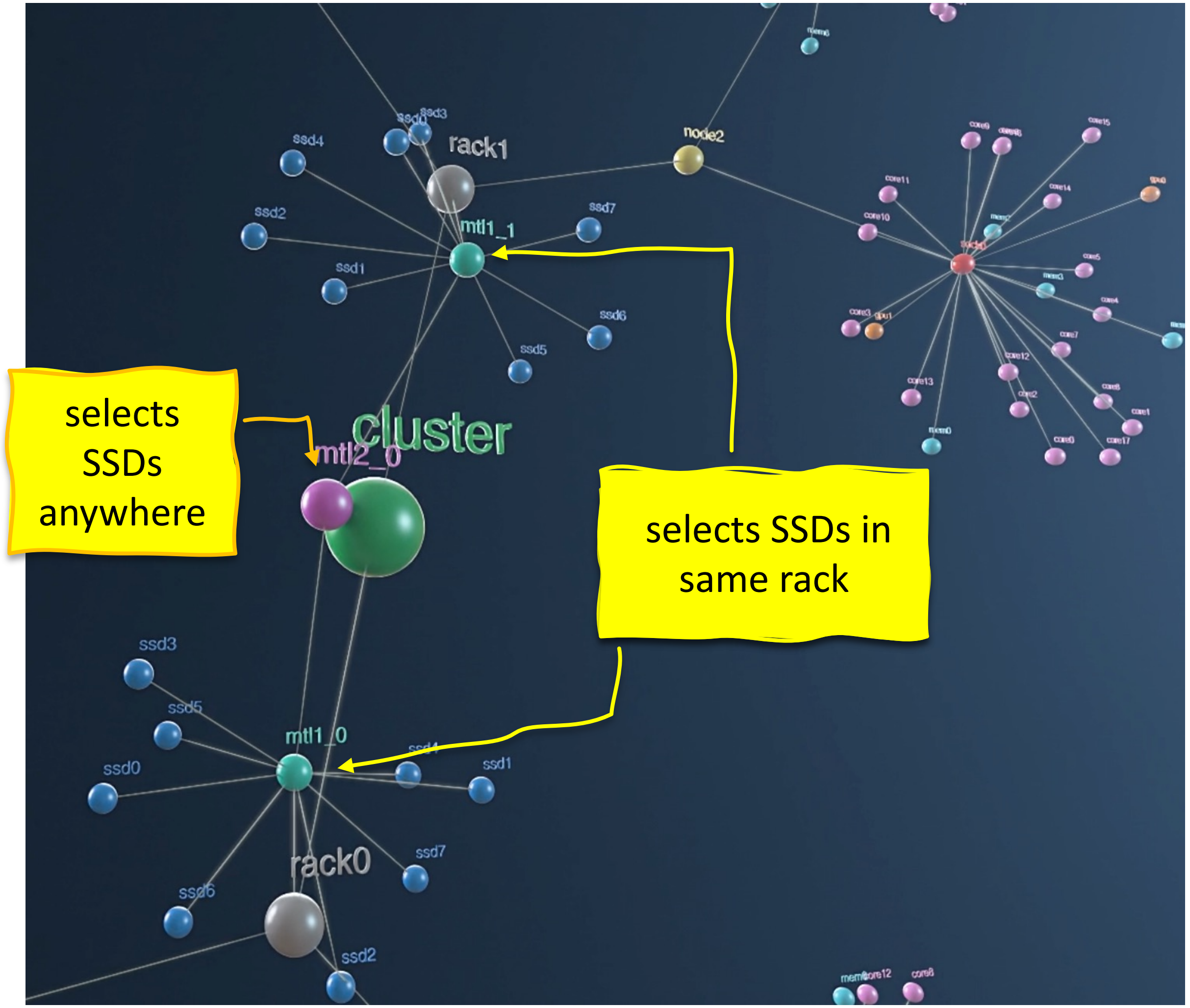
The Rabbits of El Cap: a need for sophisticated scheduling

- El Capitan will have multi-tiered storage centered around nodes called “rabbits”
  - There will be one rabbit per chassis (**N** compute nodes)
- Each rabbit node has a collection of 18 SSDs with direct PCIe connections to the compute nodes on the chassis
- The storage can be dynamically configured to offer either node-local storage or storage common to all compute nodes in a job
  - Node-local storage is connected by PCIe, global storage is over the network

The rabbits are a scheduling nightmare

- The scheduler for El Capitan needs to be aware of rabbits as both a per-rack and global resource
  - An individual rabbit can be both at once to one or more jobs
- Rabbits can be allocated independently of jobs
- There are further constraints about the number and types of storage that can be combined on a single rabbit
- Scheduling rabbits was deemed too difficult for traditional schedulers

Fluxion's graph approach can solve the rabbit scheduling problem



- In principle, Fluxion can schedule rack-local and global storage with no code change. But (full disclosure)…
- Fluxion is wasteful when it needs to schedule the same resource type multiple times
  - This affects all jobs that request multiple rabbit allocations
  - This is a known issue and is planned to be fixed before El Cap is ready
- Actual deployments currently make use of workarounds

Flux + Rabbits Deployment Status

- LLNL currently has four clusters with rabbits
  - Tioga, RZVernal, Tenaya, and Hetchy
  - 8 rabbits total
- Creating node-local and global storage works consistently, as does data movement to and from the rabbits
  - There are still some kinks to work out, especially in error handling
- Rabbits will be exposed to users soon (mid-May?)

The Fluence plugin brings HPC-grade scheduling and improved performance to Kubernetes.

K8s Scheduling Framework plugin based on Fluxion scheduler.

Architectural change from monolithic to gRPC-based

- Improves maintainability, separation of concerns

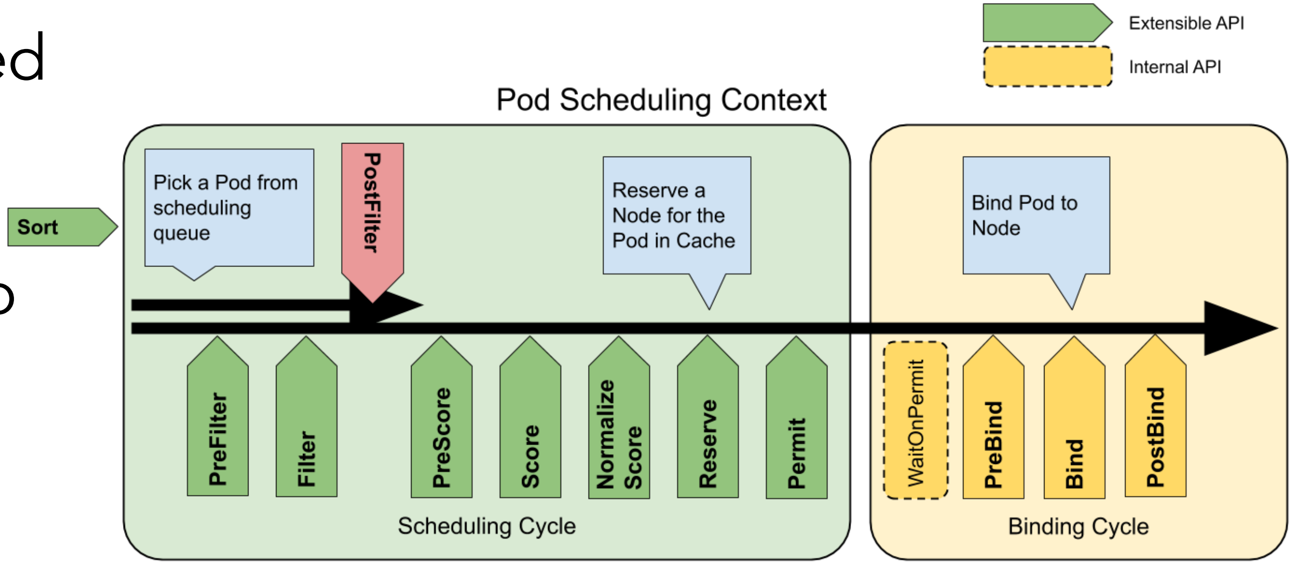


image: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

More placement control and functionality

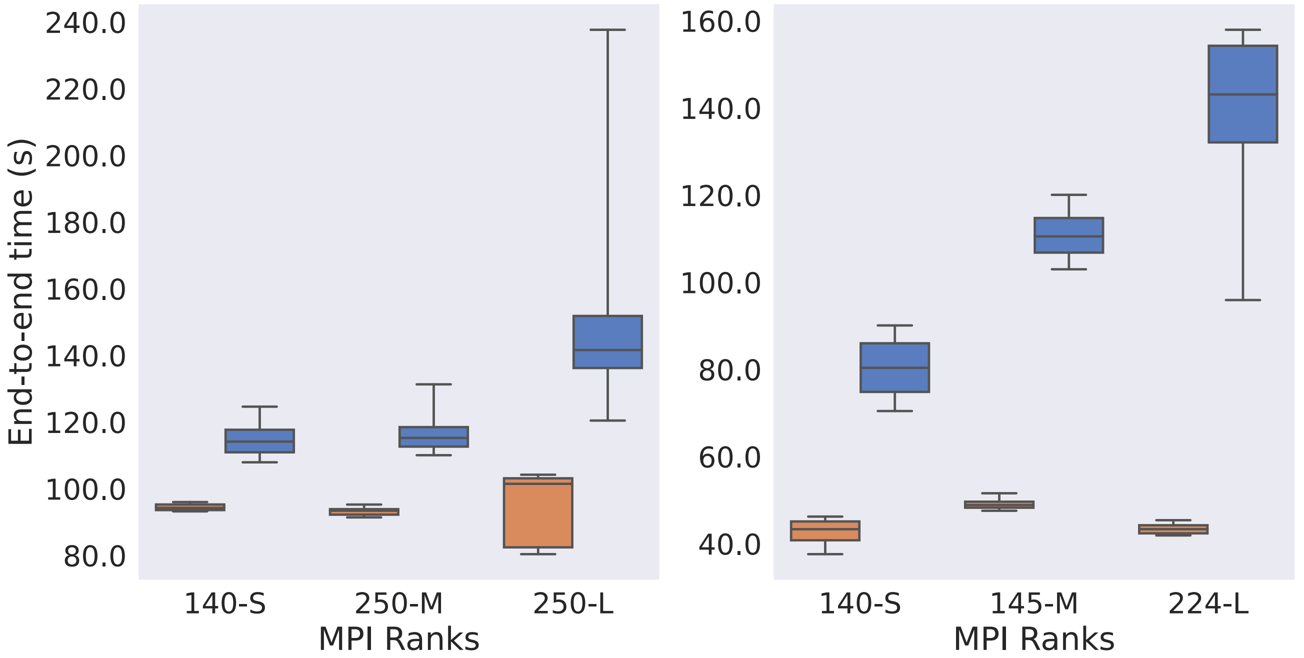
- Gang scheduling
- GPU support
- Topology awareness of Availability Zones (AZs)

Easier deployment

- Automation through Helm
- Export of Golang modules for easier distribution

Fluence accelerates simulated workflows

- Fluence-scheduler apps up to ***3.5x faster*** than Kube-scheduler
- Higher variability with kube-scheduler especially at the largest scale
- Kube-scheduler unable to pack on single node first
  - Kube-scheduler spreads pods even when limiting placement options with affinity



QMCPack <pods, ranks>
10, 140 (Small)
18, 250 (Medium)
20, 250 (Large)

LAMMPS <pods, ranks>
10, 140 (Small)
18, 145 (Medium)
20, 224 (Large)

Conclusions

Improved the MPI Operator, allowing it to scale to thousands of MPI ranks. HPC benchmarks that use MPI can scale two orders of magnitude higher than before in Kubernetes.

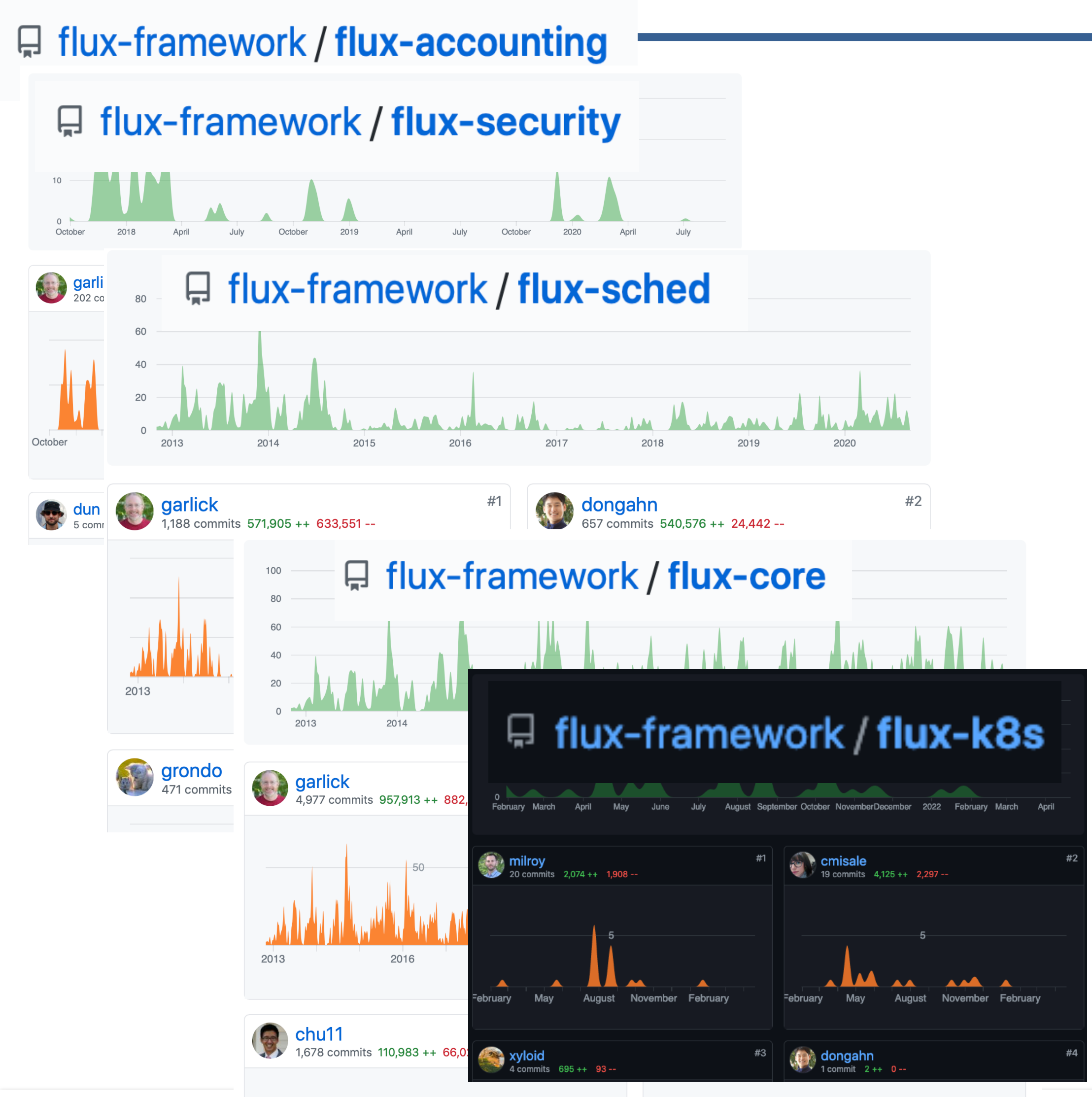
Fluence pod placement outperforms Kube-scheduler within a single Availability Zone on EKS as well as across AZs in IBM Cloud

Fluence produces deterministic placement

Kube-scheduler random tie breaking causes delayed execution for apps that exhibit dependencies between pods (MPI applications, or deployments with ***`minimum replicas`***

Kube-scheduler cannot be made to reproduce Fluence placement even with affinity and pod placement restrictions. Startup packing policy is available, user needs cluster admin privileges

Flux is a very transparent and accessible project



- Open-source project in active development at flux-framework GitHub org
  - Multiple projects: core, sched (Fluxion), security, accounting, k8s etc.
  - Over 15 contributors including some principal engineers behind Slurm
- Easily-accessible documentation and issue tracking

Links and References

- Flux framework documentation: flux-framework.readthedocs.io/en/latest/
- Documentation written by non-developers: https://hpc-tutorials.llnl.gov/flux/
- Resource manager cross-reference: https://hpc.llnl.gov/banks-jobs/running-jobs/batch-system-cross-reference-guides
- For reporting issues, asking questions, or contributing:
  - The flux-core repository: github.com/flux-framework/flux-core
  - The flux-sched repository: github.com/flux-framework/flux-sched
  - Various other repos under the flux-framework GitHub org (accounting, coral2, etc)

Thank you!
Questions?

Lawrence Livermore National Laboratory