Resource Management for Next-generation HPC Systems: Challenges and Solutions

11th Scheduling for Large-scale Systems Workshop

Tapasya Patki

May 20, 2016

LLNL-PRES-691537
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

LLNL HPC Systems

System (Program)	Processor Architecture	Nodes	Cores	Peak (TFLOP/s)
RZ				
RZCereal (M&IC)	Intel Xeon E5530	21	169	1.6
RZHasGPU	Intel Xeon E5-2667 v3	20	320	8.2
RZMerl (ASC/ M&IC)	Intel Xeon E5-2670	162	2,592	53.9
RZSLIC ***	Intel Xeon E5330	3		
RZuSeq (ASC) ****	IBM PowerPC A2	522		
RZZeus (M&IC)	Intel Xeon E5530	267		

System (Program)	Processor Architecture	Nodes	Cores	Peak (TFLOP/s)
CZ				
Ansel (M&IC)	Intel Xeon EP X5660	324	3,888	43.5
Aztec (M&IC)	Intel Xeon EP X5660	96	1,152	12.9
Catalyst (ASC/M&IC) ****	Intel Xeon E5-2695 v2	324	7,776	149.3
Cab (ASC/ M&IC)	Intel Xeon E5-2670	1,296	20,736	431.3
	AMD Opteron		256	1.6
			13,216	112.7
			40	–
			23,328	261.3
	ntel Xeon E5-2670	162	2,592	53.9
	ntel Xeon E5-2670	324	5,056	107.8
	BM PowerPC A2	24,576	393,216	5,033

System (Program)	Processor Architecture	Nodes	Cores	Peak (TFLOP/s)
Inca (ASC)	Intel X X566			
Juno (ASC)	AMD 8354			
Max (ASC)	Intel X E5-2670			
Muir (ASC)	Intel Xeon EP X5660	1,296	15,552	174.2
Sequoia (ASC) **	IBM PowerPC A2	98,304	1,572,864	20,132
Zin (ASC)	Intel Xeon E5-2670	2,916	46,656	970.4

Stats

- Max: 98,304 nodes in one system (Sequoia)
- 25 systems across open and closed zones
- Various processor architectures

Future systems present several new challenges

- Run-to-run variability, inter-job interference
- Multiple constraints: power, network, I/O, and data awareness
- Need to support high-throughput workloads, such as UQ workloads
- Increased error and failure rates

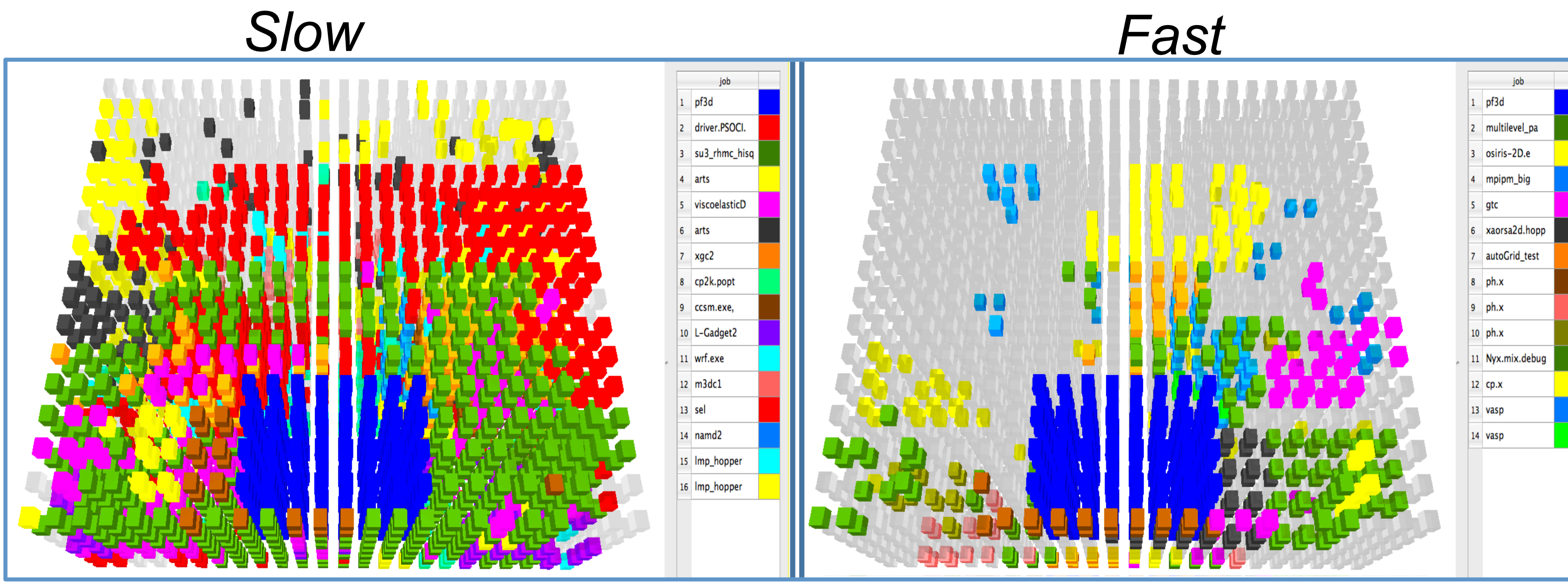
Future systems present several new challenges

- Run-to-run variability, inter-job interference
- Multiple constraints: power, network, I/O, and data awareness
- Need to support high-throughput workloads, such as UQ workloads
- Increased error and failure rates

How do we design low-overhead, scalable resource managers?

PROBLEM: Network contention and inter-job interference can lead to severe run-to-run variability

Two 512-node pF3D runs (blue) on Hopper Cray XE6



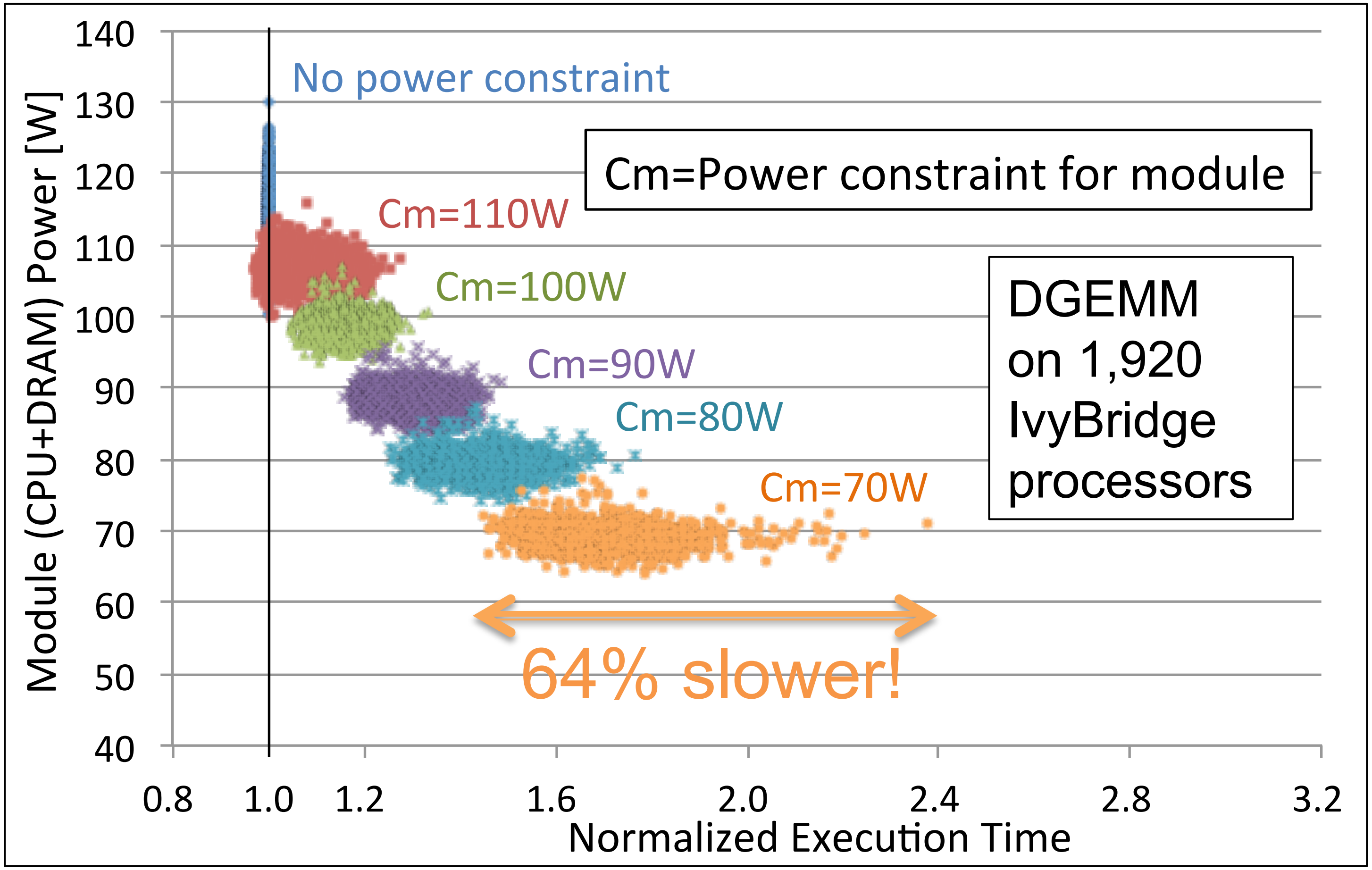
25% slower messaging rate due to other jobs!

Graph Courtesy: Abhinav Bhatele, LLNL

PROBLEM: Existing resource managers may not scale well for high-throughput ensemble workloads

- Managing several thousand jobs at once can be slow, e.g., UQ ensemble workloads
  - Job launch times vary
- Typically targeted toward fewer large-sized jobs, but not for more, small-sized jobs
- ~250 jobs on clusters that have thousands of nodes

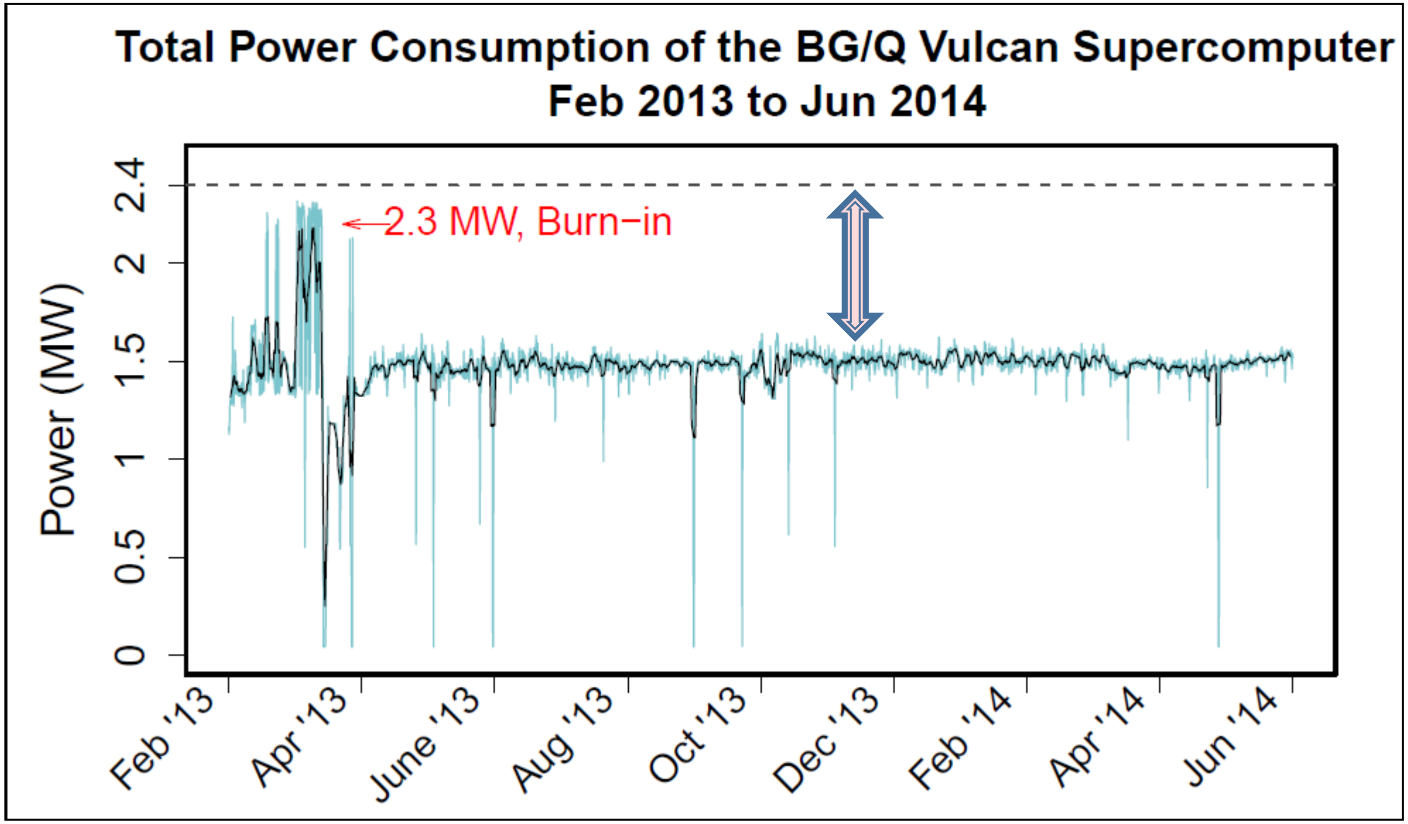
PROBLEM: Power capping affects each processor differently, creating runtime variability and imbalance



Inadomi et al., SC'15

RESULT: Unhappy Users!

UNDERUTILIZED CAPACITY: Utilizing one resource well leads to under-utilization of another resource



40% Procured Power Unused!

Patki et al., HPDC'15

Existing resource managers **cannot be extended** to support multi-constraint HPC systems easily

	Network Topology	Network Bandwidth	I/O-Aware	Power-Aware	Type?
SLURM	✔	✗	✗	⊙	Monolithic
Moab/ Torque	✔	✗	✗	✗	Monolithic
LSF, IBM	✔	✗	⊙	⊙	Monolithic
Cobalt, Argonne	✔	✗	✗	✗	Monolithic
Mesos, Apache	✔	✗	✗	⊙	2-Level Hierarchical
PBSPro, Altair	✔	✗	✗	✗	Monolithic

⊙: Limited support,
In progress

Existing resource managers are **not** designed to be fault-tolerant

- Moldable and malleable jobs are typically not supported
- Checkpoint/Restart process is slow and un-intelligent
- Users end up requesting ‘redundant’ nodes and more time as part of allocation
    - Underutilized capacity (see Felix’s talk!)

RESULT: Unhappy System Administrators!

LLNL's approach is to provide a holistic solution for a large-scale HPC system

Monitoring infrastructure for production clusters

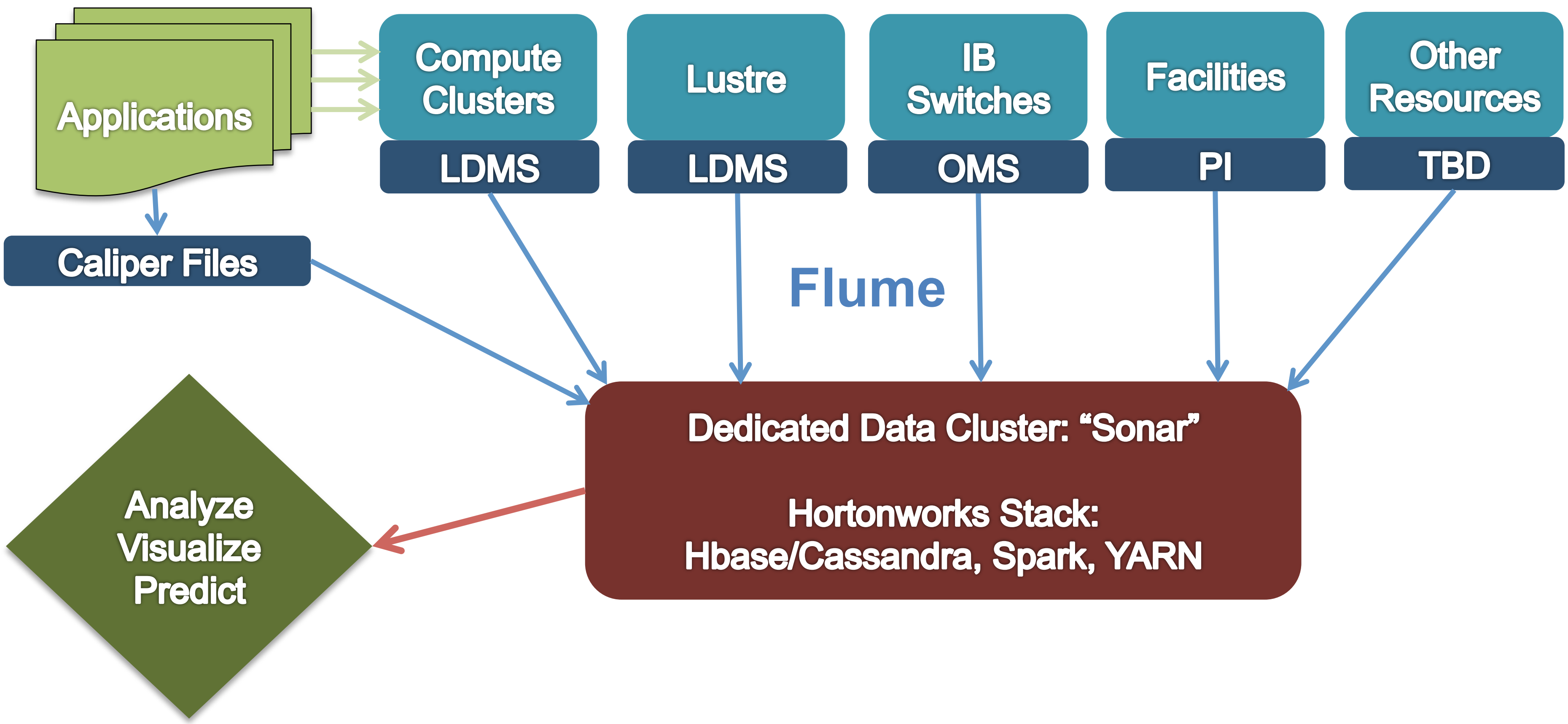
PIPER, SONAR cluster*

Framework for next-gen resource management

flux

**Performance Insights for Programmers and Exascale Runtimes*

Deploying site-wide monitoring infrastructure



PI: Todd Gamblin, LLNL

Flux Framework: Next Generation Resource Mgmt.

- Hierarchical resource manager designed to support future HPC systems in a scalable manner
- Three key components: flux-core, flux-sched, flux-capacitor
- Open source, initial release will be available soon

Flux Framework: *flux-core*

- Communication layer comprising of distributed message broker and plug-in modules for services
- Three overlay networks implemented using ZeroMQ
  - For request/response, session-wide broadcasts, and debugging
- Flux KVS/DHT module for job and resource configuration in a session
  - Useful for logging, synchronization, broadcasting

Flux Framework: *flux-sched*

- Flexible resource model to represent new hardware and flow resources
  - Accelerators
  - Power, network bandwidth
  - Easily extensible and mutable

- Scalable job management with asynchronous event-based protocols

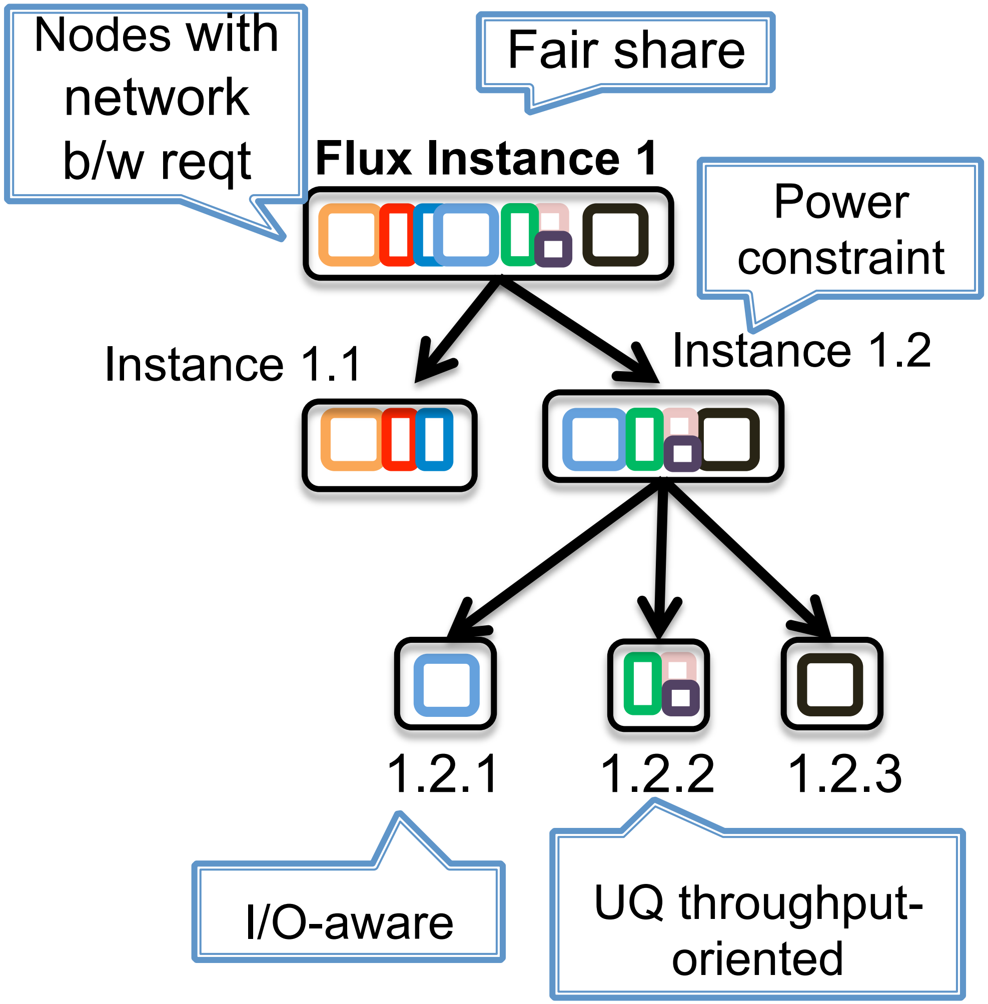
- Currently supports FCFS, easy backfilling

Flux Framework: *flux-capacitor*

- Simple python interface for high-throughput, ensemble workloads

- Supports pulsed job launch: feeds jobs (flux instances) to the system at an ideal rate

Flux Framework: Example



Flux Framework: Current Research

- Hierarchical Scheduling
  - Adds additional levels of schedulers to form a hierarchy
  - Initial study shows adding only one additional level in the reduces the scheduling complexity by 3.6x
  - But increases the resource fragmentation by up to 20%
- Dynamic Scheduling
  - Allows for allocations to change size at runtime
  - Fault-tolerance, fragmentation, utilization
- I/O Aware Scheduling
  - Efficient ways to schedule for systems with burst buffers
  - Increases system efficiency by ~1.3x in exchange for increasing turnaround time by ~1.5x

Open Research Questions

- Sources of run-to-run variability
- Analysis of user behavior, workloads, error logs
- Impact of data staging on power and network performance
- How to prioritize constraints: power, network, file systems

The Flux Team






Lawrence Livermore National Laboratory*

University of Delaware

Technische Universität Darmstadt

*And Mark Grondona, whose picture I couldn't find this morning!

Flux Resources

Project Page: flux-framework.github.io

flux-core: github.com/flux-framework/flux-core
flux-sched: github.com/flux-framework/flux-sched
flux-capacitor:github.com/flux-framework/capacitor

Lawrence Livermore National Laboratory