Fully Hierarchical Scheduling: Paving the Way to Exascale Workloads

Stephen Herbein[1,2], Tapasya Patki[2], Dong H. Ahn[2], Don Lipari[2],
Tamara Dahlgren[2], David Domyancic[2], Michela Taufer[1]
[1]University of Delaware, [2]Lawrence Livermore National Laboratory

Motivation

- Emerging HPC workloads represent an order of magnitude increase in both scale and complexity; yet batch scheduling remains stuck in the decades-old, centralized scheduling model

> "It is widely expected that rigorous **uncertainty quantification** over high-dimensional input spaces will play a crucial role in enabling **extreme-scale science.** Indeed, a thousand-fold increase in computing power would facilitate **orders-of-magnitude more simulation** realizations"

From the Top Ten Exascale Research Challenges. DOE ASCAC Subcommittee Report. 2014.

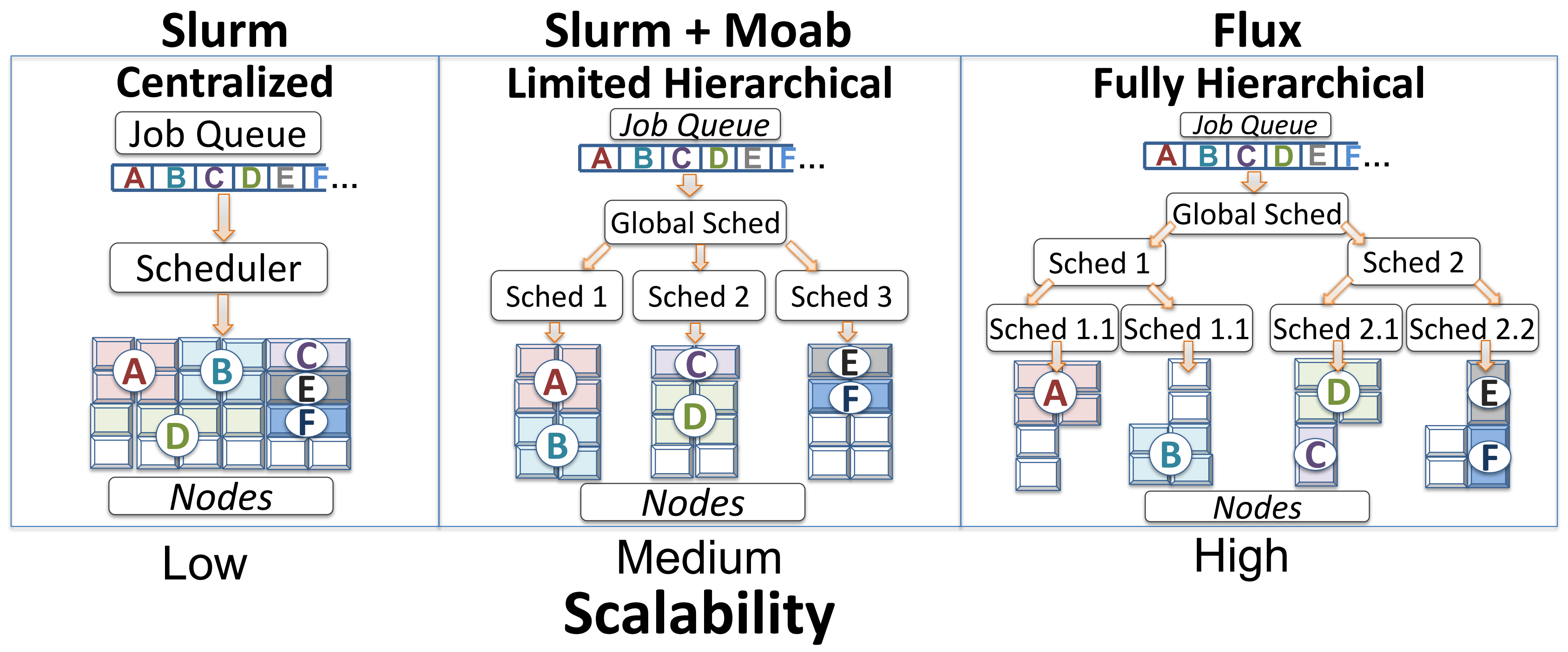
- HPC batch schedulers have several limitations with respect to these emerging workloads, which has led to a proliferation of workflow systems that provide specialized workarounds [2,3]

Schedulers' Limitations	Workload System's Workaround	Side Effects
Max number of jobs	Throttle submissions	Decreased job throughput
Limited job throughput	Aggregate jobs	Increased workload runtime
Lack of job/ensemble status & control API	Track individual job's status through files	I/O bottleneck
Lack of programmable failure detection	Inspect failures manually	Unnecessary job resubmissions

The fully hierarchical scheduling model and its implementation, Flux, provide general solutions to these limitations

Fully Hierarchical Scheduling Under Flux

- New HPC scheduling model aimed at addressing next-generation scheduling challenges using **one common resource and job management framework** at both system and application levels [1]
- Applies a divide-and-conquer approach to scheduling, allowing for the distribution of scheduling work across an arbitrarily deep hierarchy of schedulers



Slurm — Centralized: Job Queue (A B C D E F…) → Scheduler → Nodes. Low

Slurm + Moab — Limited Hierarchical: Job Queue (A B C D E F…) → Global Sched → Sched 1, Sched 2, Sched 3 → Nodes. Medium

Flux — Fully Hierarchical: Job Queue (A B C D E F…) → Global Sched → Sched 1 (Sched 1.1, Sched 1.1), Sched 2 (Sched 2.1, Sched 2.2) → Nodes. High

Scalability

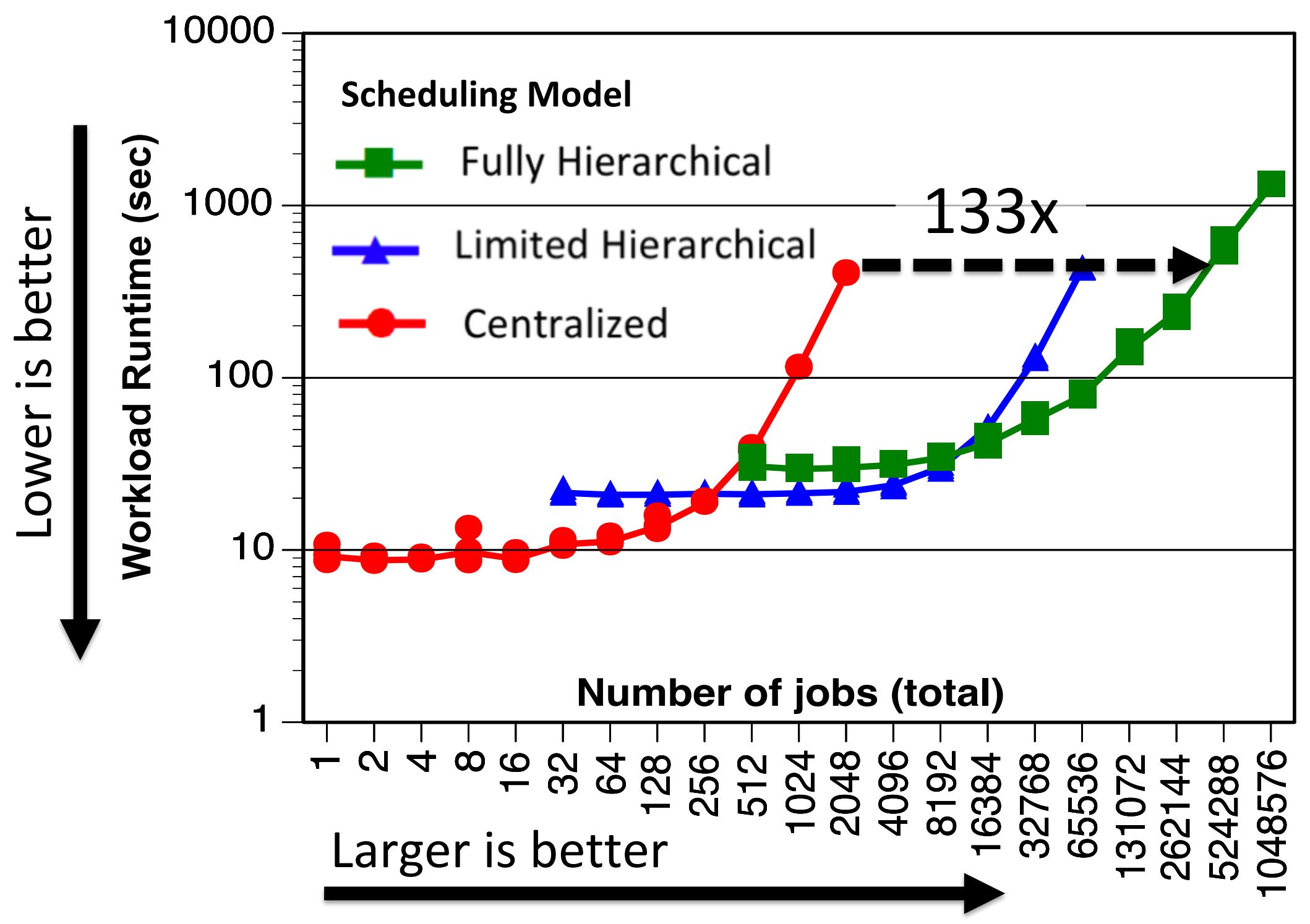
Case Study: Synthetic Stress Test

- Study configuration: all three scheduler models evaluated on a 32 node cluster with a synthetic workload of dummy jobs

Schedulers' Limitations on Number of Jobs

Centralized Model
- Exhausts local resources when handling large numbers of jobs [2]

Fully Hierarchical Model
- Distributes local resource requirements across multiple schedulers



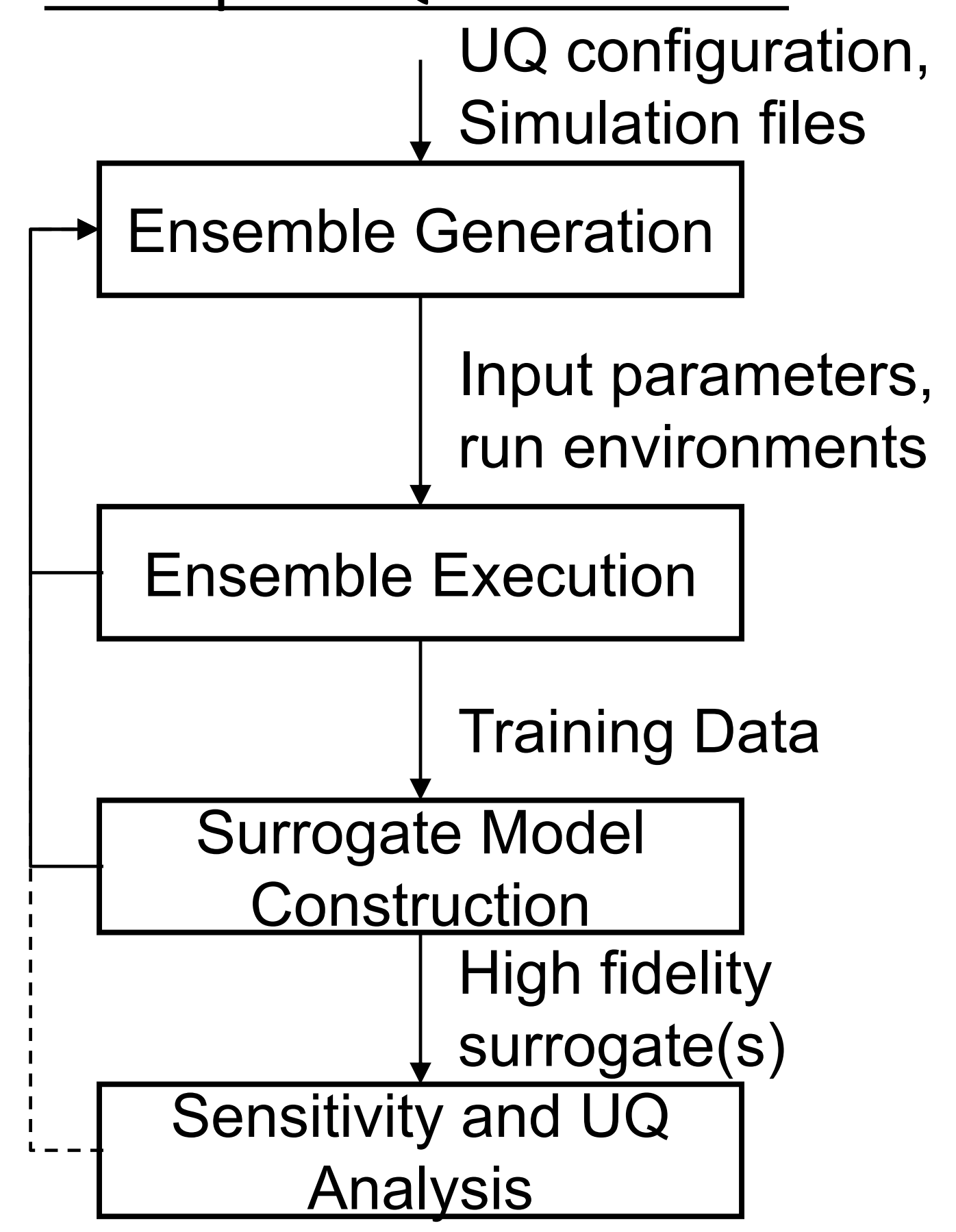
Moving from the centralized to the fully hierarchical model increases the scheduler's job scalability by 133x

Uncertainty Quantification Pipeline (UQP)

- Accounts for roughly 50.9 million CPU hours each year at LLNL
- Simplifies performing uncertainty quantification studies
- Requires running an ensemble of simulations containing anywhere between 1,000 and 100,000,000 jobs
- Provides workarounds for existing HPC schedulers' limitations
  - Workarounds result in decreased job throughput and an I/O bottleneck

The fully hierarchical scheduler handles all of the challenges encountered by the UQP

Example UQP Workflow



UQ configuration, Simulation files → Ensemble Generation → Input parameters, run environments → Ensemble Execution → Training Data → Surrogate Model Construction → High fidelity surrogate(s) → Sensitivity and UQ Analysis

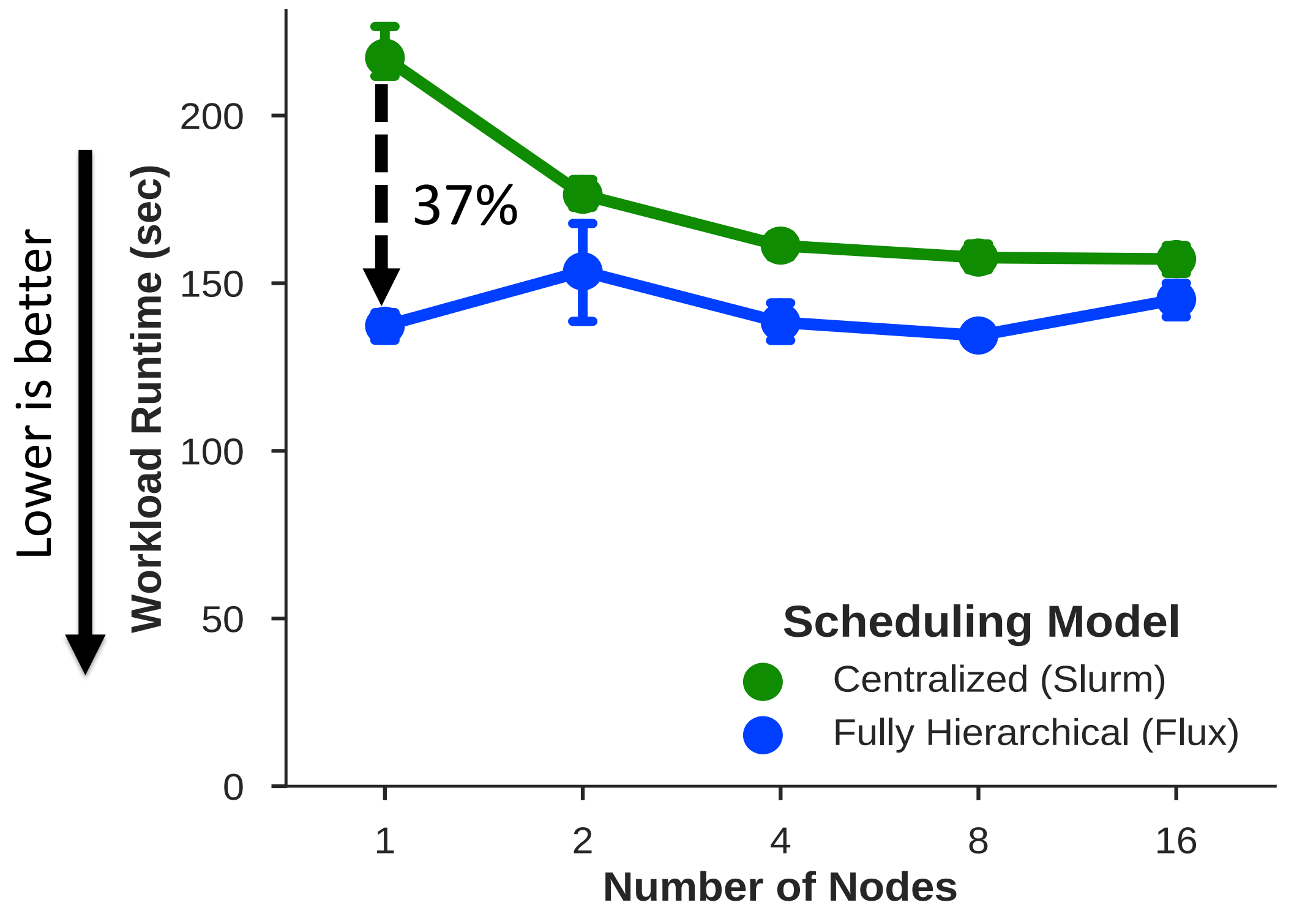
Case Study: UQP Workload

- Study configuration: UQP runs with Slurm and Flux evaluated on a 16 node cluster with a workload of a single-core Monte Carlo application [3]

Schedulers' Limitations on Job Throughput

Centralized Model
- Tasks within an aggregated UQP job are run serially, increasing the workload's runtime

Fully Hierarchical Model
- Aggregated job is managed by its own full-featured scheduler, allowing tasks to be run concurrently

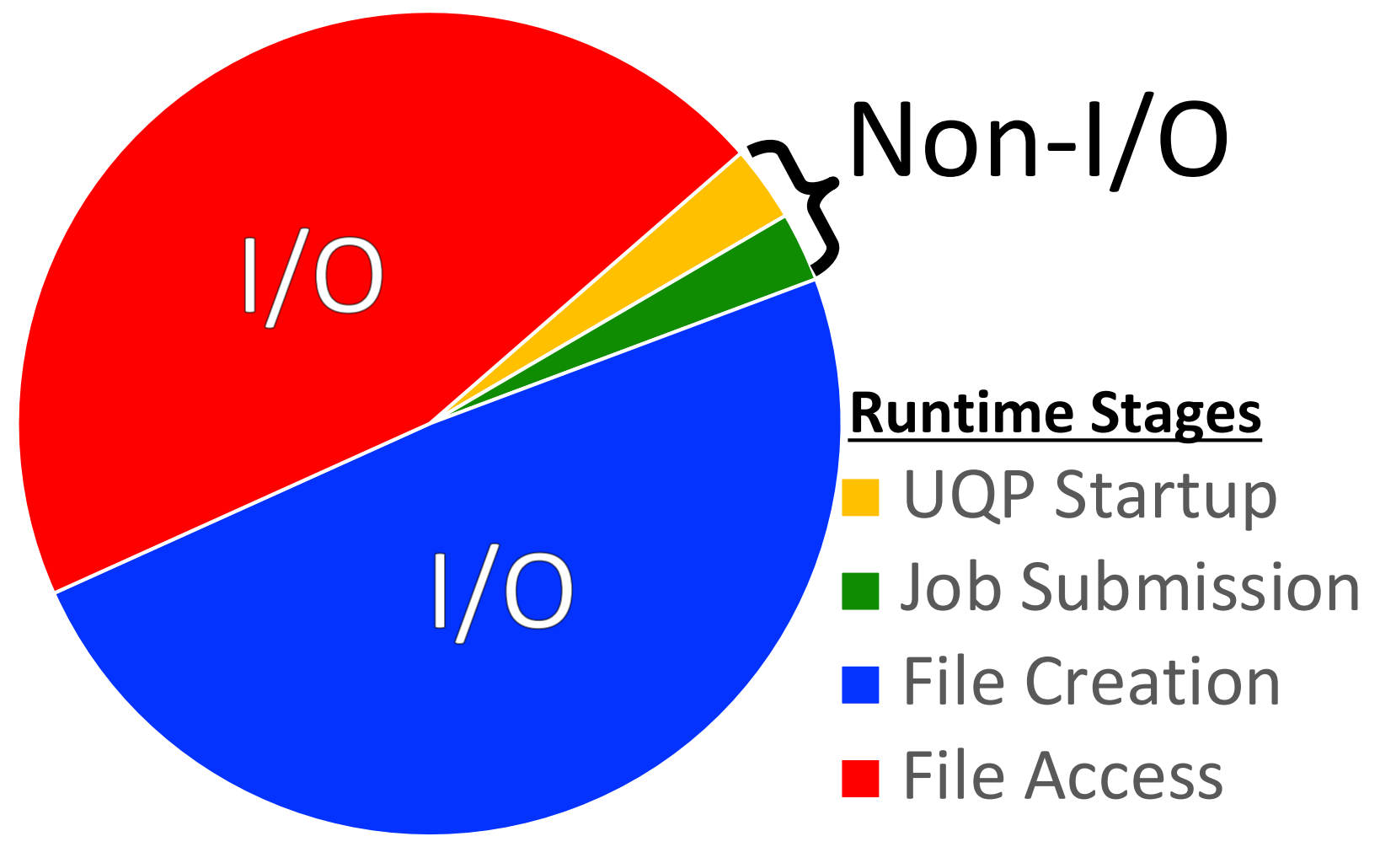


Moving from the centralized scheduler Slurm to a fully hierarchical scheduler Flux results in a 37% faster workload runtime

Schedulers' Limited Job Ensemble Support

Slurm (Centralized Scheduler)
- Limited API for job status
- UQP tracks job states through files, creating an I/O bottleneck

Flux (Fully Hierarchical Scheduler)
- Provides a subscription-based job status API, eliminating I/O



Centralized Model
- Submit and track each job individually

Fully Hierarchical Model
- Submit hierarchies of jobs and track them at variable levels of granularity through an API

Flux and the fully hierarchical model simplifies the submission and tracking of job ensembles and thus eliminates the need for the UQP's I/O

Future Work

- Integrate Flux's job/ensemble status & control API into the UQP to simplify the submission/tracking of the job ensembles while also eliminating the I/O bottleneck
- Develop a programmable failure detection mechanism within Flux to reduce unnecessary resubmissions and simplify error handling for users

References and Acknowledgements

[1] D. Ahn, et. al. Flux: A Next-generation Resource Management Framework for Large HPC Centers. In ICCPW'14.
[2] J. Gyllenhaal, et. al. Enabling High Job Throughput for Uncertainty Quantification on BG/Q. In ScicomP'14.
[3] T. Dahlgren, et. al. Scaling Uncertainty Quantification Studies to Millions of Jobs. In SC'15.

- The authors acknowledge the advice and support from the Flux team and others at LLNL: Tom Scogland, Jim Garlick, Mark Grondona, Becky Springmeyer, Chris Morrone, and Al Chu.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. National Science Foundation CCF #1318445. This research was supported by the Exascale Computing Project: 17-SC-20-SC