Exploring the Trade-off Space of Hierarchical Scheduling for Very Large HPC Centers

Stephen Herbein[1,2], Dong H. Ahn[2], Don Lipari[2], Tom Scogland[2], Kento Sato[2], Jim Garlick[2], Mark Grondona[2], Becky Springmeyer[2], Michela Taufer[1]

[1]University of Delaware, [2]Lawrence Livermore National Laboratory

Motivation

- On the path to a next-generation HPC center lies a significant increase in scale and diversity of resources
- Single, monolithic scheduling already lacks scalability and flexibility for today's large HPC centers
- Hierarchical scheduling is becoming an attractive alternative
  - A lack of trade-off studies precludes the development of effective techniques

	2010	2018	Change
System Size (nodes)	20 K nodes	1 M nodes	50x
Power	6 MW	20 MW	3x
Storage	15 PB	300 PB	20x
Interconnect BW	1.5 GB/s	50 GB/s	33x
Total Concurrency	225 K	1 B	4,444x

DOE Exascale Initiative Roadmap, Architecture and Technology Workshop, San Diego, December, 2009.

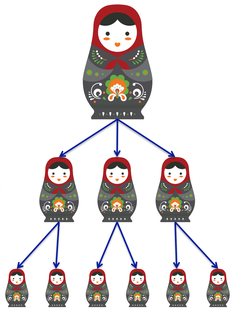
We need systematic trade-off studies to be able to develop effective hierarchical scheduling techniques

Goal

- Quantify the advantages and disadvantages of a hierarchical, multilevel scheduling scheme against a monolithic scheme
- In particular, explore the trade-off space between scheduling complexity and resource utilization

Hierarchical Scheduling under Flux

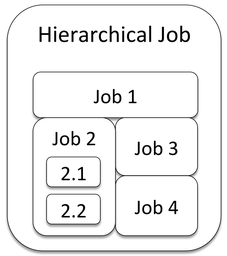
- Flux is a resource and job management system (RJMS) currently being developed to enable hierarchical, multi-level scheduling for large HPC centers [1]
- Its hierarchical scheduling rules:
  - *Parent bounding rule* – parent grants and confines the allocation of its children
  - *Child empowerment rule* – children are solely responsible for the most efficient use of their resources
- Major scalability and policy requirements
  - Provide higher levels of scheduler parallelism and thus scalability
  - Distribute load across the schedulers in this hierarchy
  - Provide the ability to impose a stricter policy enforcement



Source: eftours.com/worldmap

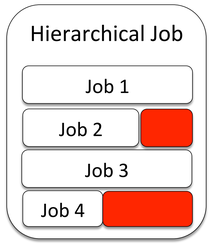
Realistic Hierarchical Workload Creation

- Under hierarchical scheduling, any job can instantiate a scheduler to schedule its sub-jobs
- A hierarchical workload is non-existent to be used for our trade-off exploration
- Use novel job-aggregation techniques to generate hierarchical workloads from real HPC workloads
  - Jobs submitted within a short window of time with similar characteristics are aggregated together into a larger job
  - This emulates an important mode in which users will use hierarchical schedulers under Flux

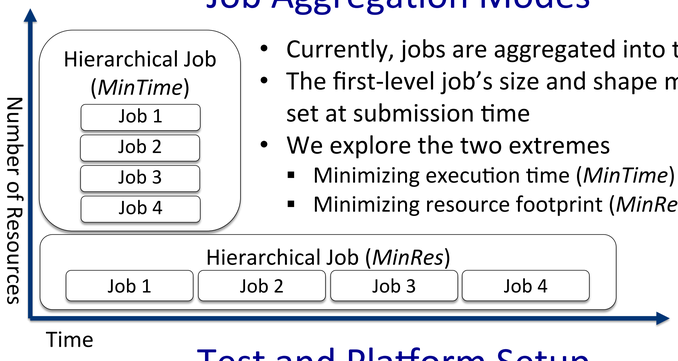


Metrics

- Scheduling complexity
  - Is a function of the number resources items and jobs to be considered at each level in the hierarchy
  - Increases as compute resources increase in scale and diversity
  - Decreases when the number of jobs are reduced by job aggregation at each level in the hierarchy
- Resource utilization
  - Hierarchical scheduling can decrease utilization when resources within an allocation go idle
  - Typically caused by small or short-running jobs



Job Aggregation Modes



- Currently, jobs are aggregated into two levels
- The first-level job's size and shape must be set at submission time
- We explore the two extremes
  - Minimizing execution time (*MinTime*)
  - Minimizing resource footprint (*MinRes*)

Test and Platform Setup

- Evaluate with a scheduling emulator built on top of Flux
- Use job logs from Rzmerl & Rzzeus, clusters at LLNL, as input to the emulator
  - Logs were collected over 2.5 months and contain 32,046 jobs
- Aggregate jobs that are submitted by the same user and within 30 seconds of each other

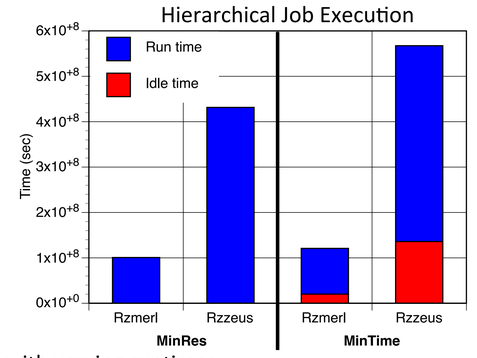
Scheduling Complexity Results

Machine	Pre-Aggregation	Post-Aggregation	Ratio
Rzmerl	6,737 jobs	5,688 jobs	1.18x
Rzzeus	25,309 jobs	7,073 jobs	3.58x

- Introducing only one additional level to scheduling leads to a reduction of **3.58x** in scheduling complexity for Rzzeus workloads
  - The preliminary results show each additional level will become a significant reduction factor for scheduling complexity
- Rzmerl has a lower ratio since its submission rate is less
  - Lower probability that a user submits jobs back-to-back

Resource Utilization Results

- *MinRes* results in virtually no idle time and thus no decrease in resource utilization compared to monolithic scheduling
  - Idleness caused by sub-jobs with varying sizes
- *MinTime* results in a large decrease in utilization compared to monolithic scheduling
  - Idleness caused by sub-jobs with varying runtimes



Conclusion and Future Work

Hierarchical scheduling offers a sizable reduction in scheduling complexity

- Job submission patterns suggest users can group together their jobs to take advantage of hierarchical scheduling
- Larger and busier HPC centers can reduce their scheduling complexity with hierarchical scheduling at deeper levels
- Sub-jobs with diverse time and resource requirements can leave center resources underutilized
- Our study motivates the development of dynamic scheduling as a way to complement hierarchical scheduling

References and Acknowledgements

[1] D. Ahn, J. Garlick, M. Grondona, D. Lipari, B. Springmeyer, and M. Schulz. Flux: A next-generation resource management framework for large hpc centers. In Proceedings of the 43rd International Conference on Parallel Processing Workshops, ICCPW'14, Sept 2014.

The authors acknowledge advice and support from the Flux team and others at LLNL: Don Lipari, Tom Scogland, Kento Sato, Jim Garlick, Mark Grondona, and Becky Springmeyer. The authors are grateful to Adam Moody and Todd Gamblin for access to the LLNL cluster job logs.

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

LLNL-POST-675766