Slurm Simulator Development: Balancing Speed, Accuracy, and Maintainability

Nikolay A. Simakov[1] (PI), Robert L. DeLeon[2] (Co-PI)
[1]nikolays@buffalo.edu,[2]rldeleon@buffalo.edu

Abstract

Slurm is an open-source job scheduling system widely used in many high-performance computing (HPC) resources. A Slurm simulator facilitates parameter tuning to optimize throughput or meet specific workload objectives. In the previous simulator version (v2) [3], the priorities were to minimize the changes to core Slurm and have a high simulation accuracy. This resulted in speed-dependent accuracy and a simulation speed only 20-40 times faster than real-time (for a midsized system). This is not a very practical simulation speed, and it is more beneficial to trade some accuracy for increased speed. The expectation was that with diminished absolute accuracy, we could still make reasonable relative predictions.

To achieve the desired speed-up goal, we use the same strategy as in our original Slurm simulator (v1) [1,2] namely: serialize the code and call all Slurm functions from a single thread in an event-driven fashion. Our simulator's resulting version (v3) has more than 500 times acceleration over real-time, allowing simulation of a month-long workload in 90 minutes.

The simulator was tested on a Mid-sized System containing 216 heterogeneous nodes containing a mixture of resources (two types of regular compute nodes, large memory nodes, and GPU nodes). The workload (also known as job traces) was based on the historical workload at our center and consisted of almost 30,000 jobs. It requires more than 29 actual days to be executed. The reference data was obtained using our Virtual Cluster, where each cluster node is represented with its own container and has a normal Slurm installed on it (see [3] for more details). To estimate the ability to predict relative values rather than absolute ones, we also vary the priority factor of several QoS groups (priority and supporters) while keeping the general QoS the same.

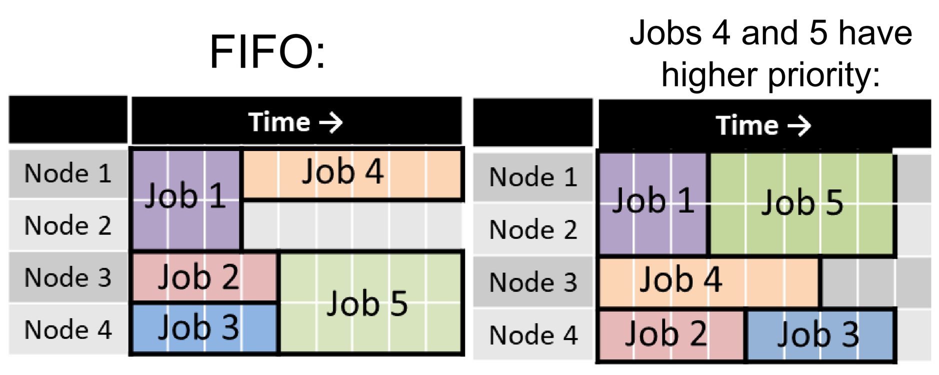
The scheduling in Slurm is a stochastic process [1,2], which has a high manifestation on highly utilized systems. Therefore, it is crucial to have a sufficient number of independent runs. It is easy to obtain multiple runs with the Slurm simulator as it is several hundred times faster than real-time; however, Virtual Cluster goes only as fast as real-time, and it takes a lot of time to get through 29 days of the test workload.

For a Mid-sized system, we found that the absolute value of mean wait time differs between Virtual Clusters and Slurm Simulator. However, the trend is very similar, especially on higher values of mean wait time.

Introduction

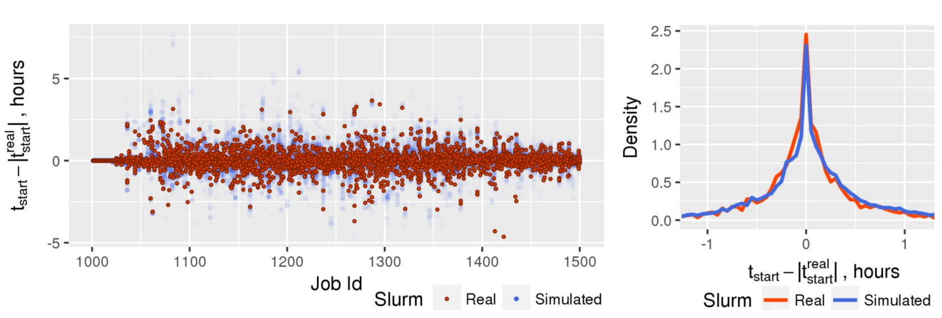
- Slurm is an open-source resource workload manager for HPC systems
  - It provides high configurability for heterogeneous resources and job scheduling
  - It is used on a large range of HPC resources from small to very large systems.
  - All current and large portion of ACCESS-CI HPC resources uses SLURM

Computational jobs can be scheduled differently:



Changing configuration on live system can have unintended adverse consequences
- It is also often hard to judge the effect

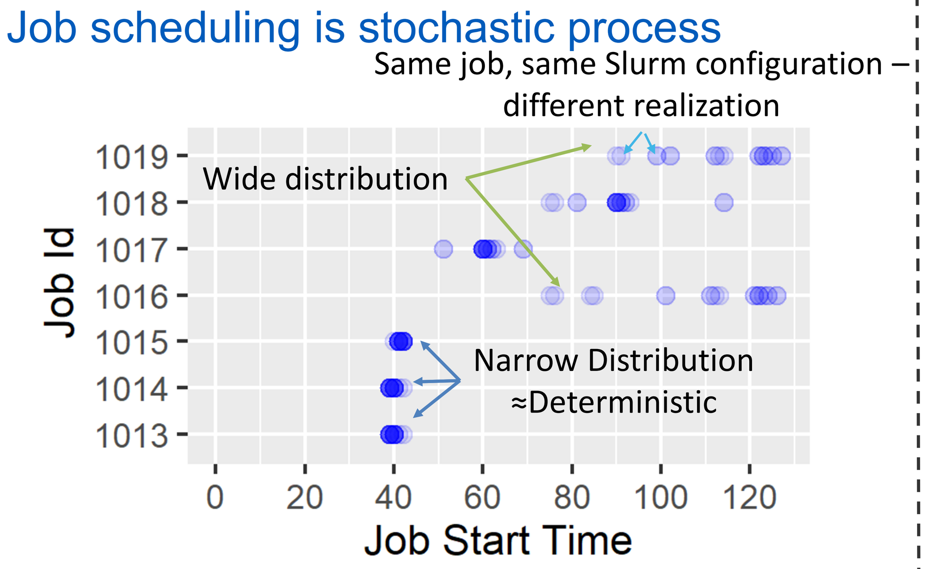
Why do We Need a Slurm Simulator?
- Finding the most optimal parameters for a Slurm deployment
- To check Slurm configuration prior to deployment
- Future system modeling
- Obtain results faster than a real time!

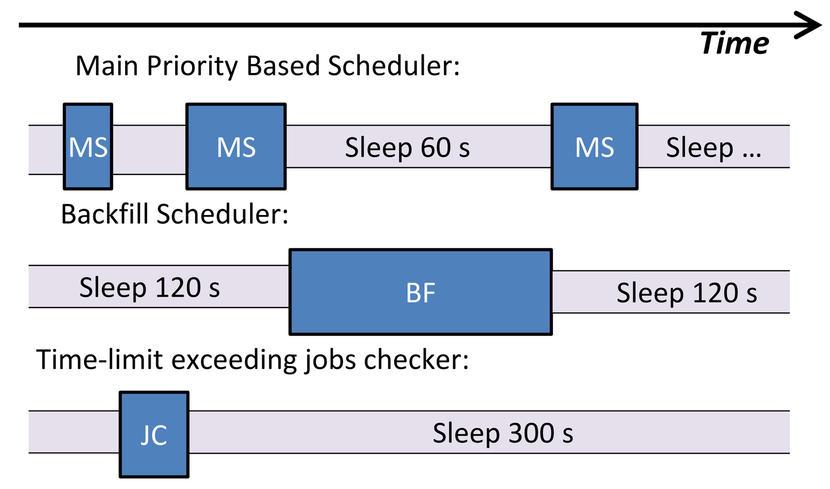


In our previous version (v2):
- The approach to converting Slurm to a Slurm simulator was to maintain a multi-thread design, minimize the changes to core Slurm (to improve code merging capabilities), and have a high simulation accuracy.
- Unfortunately, it resulted in a simulator for which accuracy was simulation-speed dependent, and speed itself was not very high. For the midsized system (217 nodes), it was in the range of 20-40 times faster than real-time. That is a month-long workload simulation is done in one day.
- This is not a very practical speed, given that you need a number of runs to generate a statistically significant result.

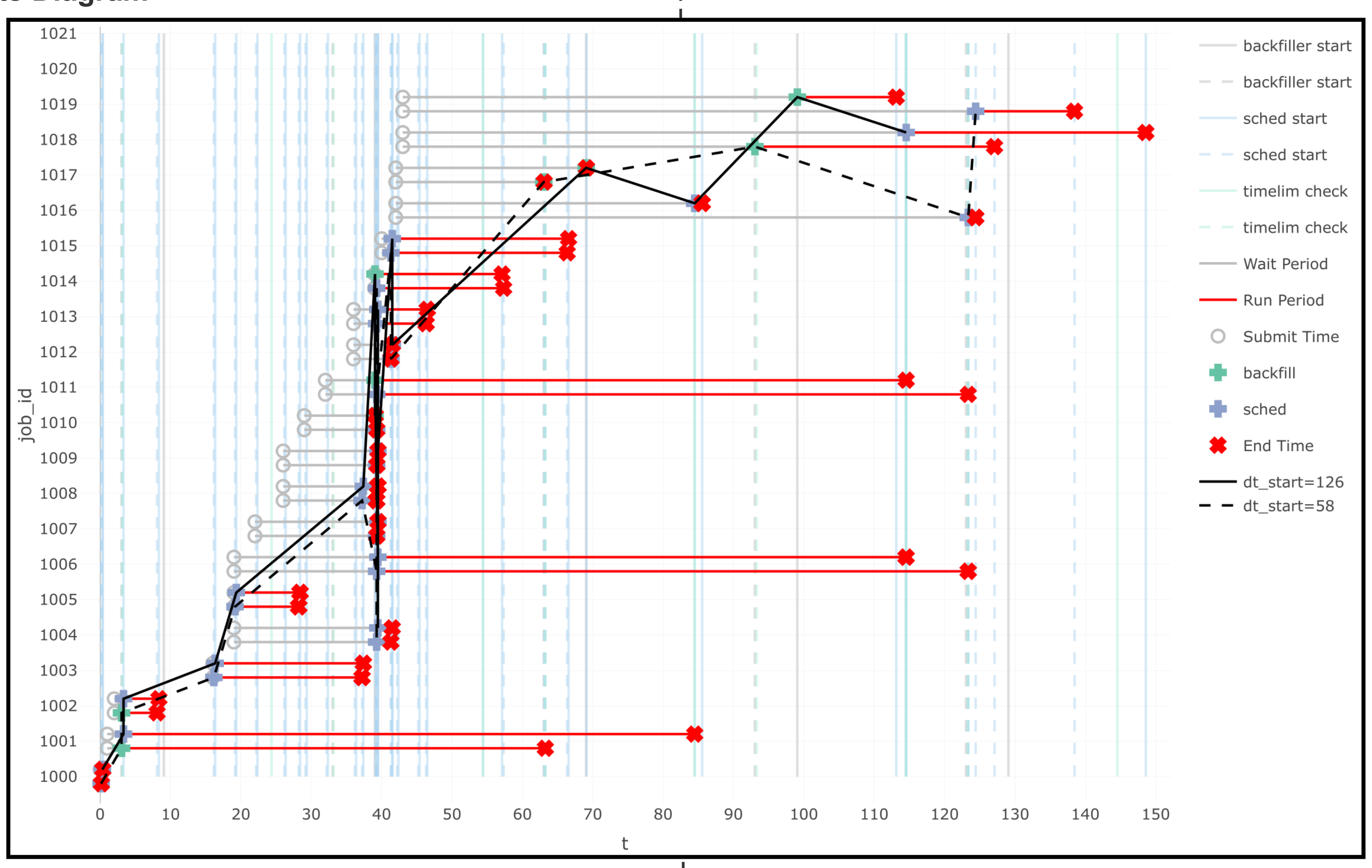
It would be more practical to trade some accuracy for the increased simulation speed. The goal of this round of developments is a good speed-up with reasonable accuracy.

Job scheduling is stochastic process





Events Diagram



Challenges with Slurm Simulation

- Job scheduling is stochastic process
- Several scheduling related routines are executed in **aperiodic** manner and **asynchronously** between each other
- In general, the location of user's job submission time around these routines is **uncertain**.
- Jobs starting time in same workload are dependent due to competition for same resources
- Single workload realization is one multidimensional data-point
- We need multiple workload realization (Slurm runs

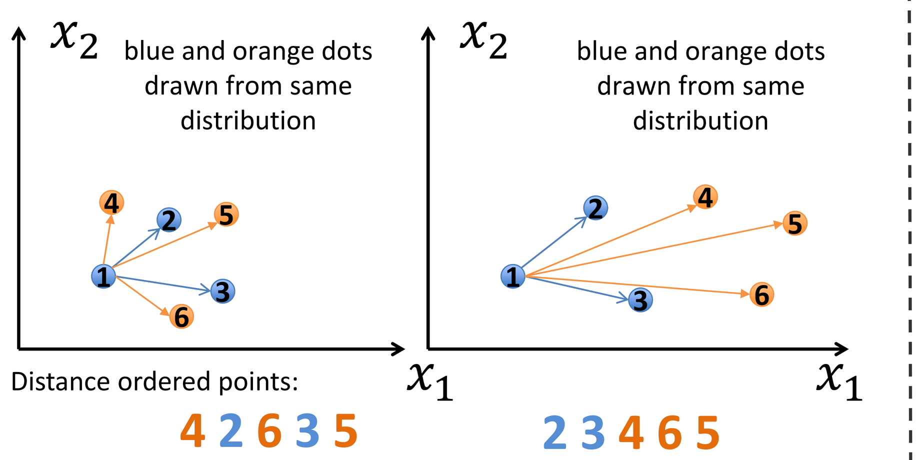
Methods

Developing Accurate Slurm Simulator
1. Start from most resent Slurm version,
2. Apply changes
3. Check that it still produce same result as reference workload realizations
4. Repeat 2 and 3 until you got fast and accurate Slurm Simulator

Multiple reference workload realizations is calculated with Virtual Clusters and normal Slurm installation
- Virtual Clusters implements HPC clusters using Docker containers
- One container per head node and each compute node
- Regular Slurm is installed

Comparing two realization:
- For short workloads Virtual Cluster and Slurm simulator can produces identical realization
- Using events diagram
- Using Euclidian distance between two runs. If w_{ij} is a wait time for j-th job in i-th realization, then the Euclidian distance between i-th and i'-th realization is $d_{ii'} = \sqrt{\sum_{j=1}^{n}\left(w_{ij} - w_{i'j}\right)^2}$

Compare multiple realization:
- Using Euclidian distance between multiple runs, plot it as a heatmap
- For hypothesis testing multivariate Wilcoxon rank sum test is used



New Slurm Simulator

- Based on Slurm 23.02
- To achieve the desired speed-up goal, we use the same strategy used in our original Slurm simulator (v1), namely: serialize code and call all Slurm functions from a single thread within an event-driven fashion (example of events: submit job, start job, run backfill scheduler). Serial execution removes the time spent in thread locks, and event-based simulation allows more aggressive time skipping.
- To minimize Slurm core code modifications, we used several approaches to alter the Slurm execution:
  - used the GCC compiler option to wrap standard library functions (--wrap option for gettimeofday, sleep and so on)
  - used the constructor method to initialize the simulator and thus avoid modifications to the main functions and
  - to access static variables and functions we created a wrapper for the source code with additional functionality to access variables and functions of interest.
- Time handling:
  - Normally time progress with regular speed
    - it is possible to scale time to model faster or slower hardware
  - Opportunistic time skipping is done by calculating the time to the next event, if the time is within a threshold the clock is incremented
- Text-based specification for batch jobs extends *sbatch* format:

```
-dt 1 -e submit_batch_job | -J jobid_1002 -sim-walltime -1 --uid=user1 \
    -t 00:01:00 -n 1 --ntasks-per-node=1 -A account1 -p normal -q normal \
    --constraint=CPU-N pseudo.job
-dt 2 -e submit_batch_job | -J jobid_1003 -sim-walltime 5 --uid=user4 \
    -t 00:01:00 -n 1 --ntasks-per-node=1 -A account2 -p normal -q normal \
    --mem=500000 pseudo.job
```

Test Systems and Reference Workload

Micro Cluster (10 Nodes)

Specially designed to test different aspects of job placement
- 4 Compute Node with CPU-N
- 4 Compute Node with CPU-M
- 1 GPU Node with 2 GPUs
- 1 Large Memory Node

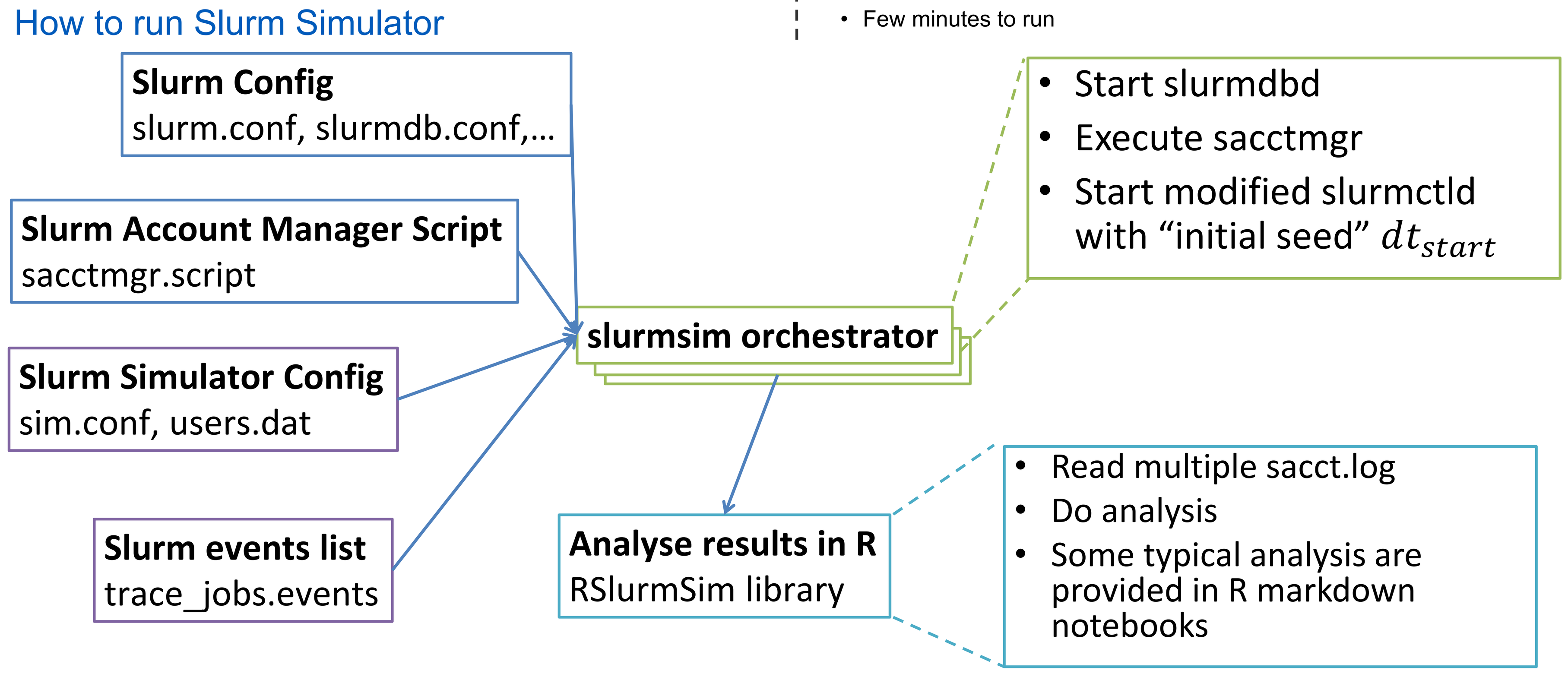
Workloads:
- 500 jobs: 500 jobs, 5 users, 2 accounts
  - 13.7 hours to run on Virtual Cluster
  - 20 runs were performed
- 500 jobs shrunken:
  - 48 minutes to run on Virtual Cluster
- Small: 20 jobs
  - Few minutes to run

UB-HPC Cluster (217 nodes)

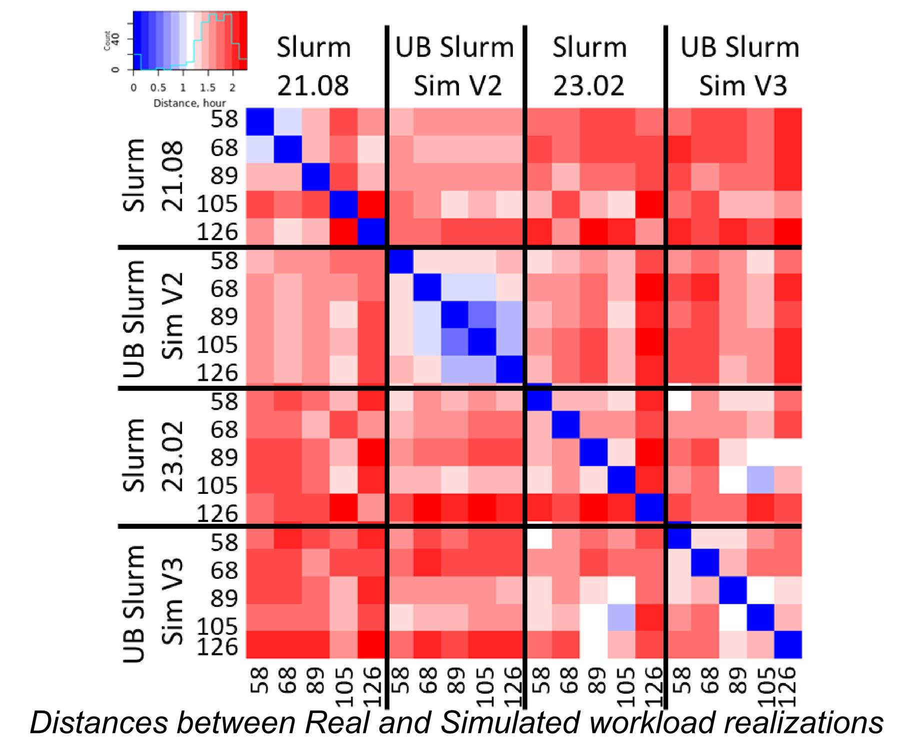
Modelled after newer part of our center academic HPC cluster
- 32 cores per node nodes:
  - 87 Compute nodes
  - 2 GPU nodes
- 40 cores per node nodes:
  - 96 Compute nodes
  - 8 GPU nodes
  - 24 large memory nodes

Workload
- Based on historic workload and contains 29,678 jobs from 95 user among 65 groups.
- **29.4 days per run** on Virtual Cluster
- 8 runs were done.

How to run Slurm Simulator



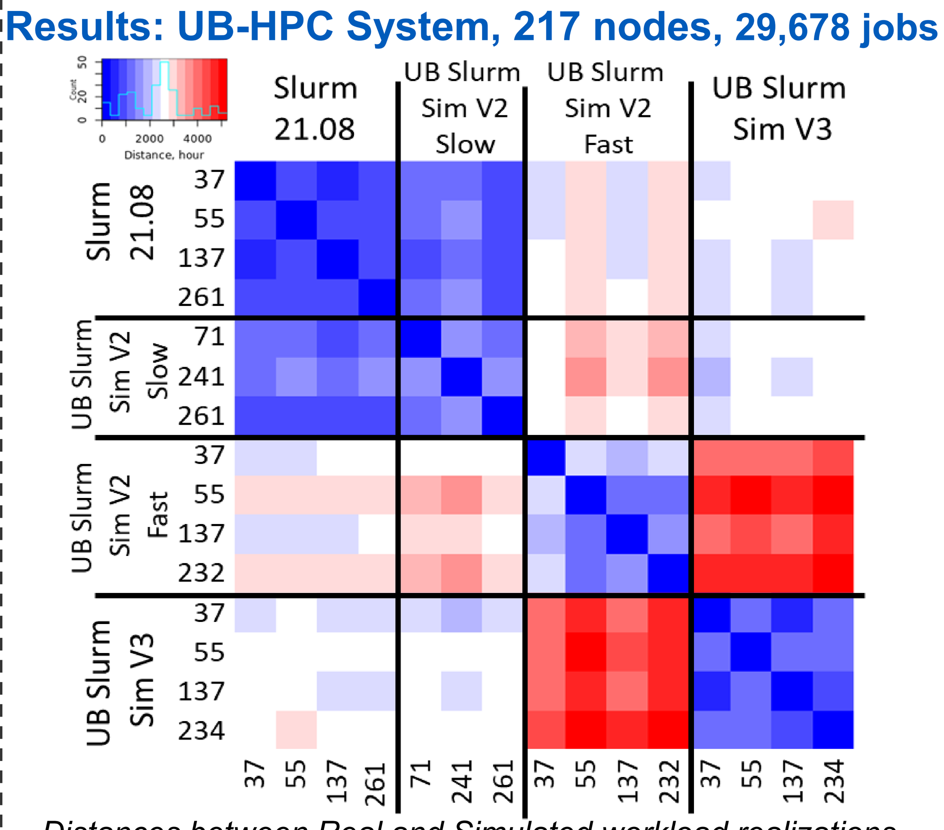
Results: Micro System, 500 job workload



Distances between Real and Simulated workload realizations

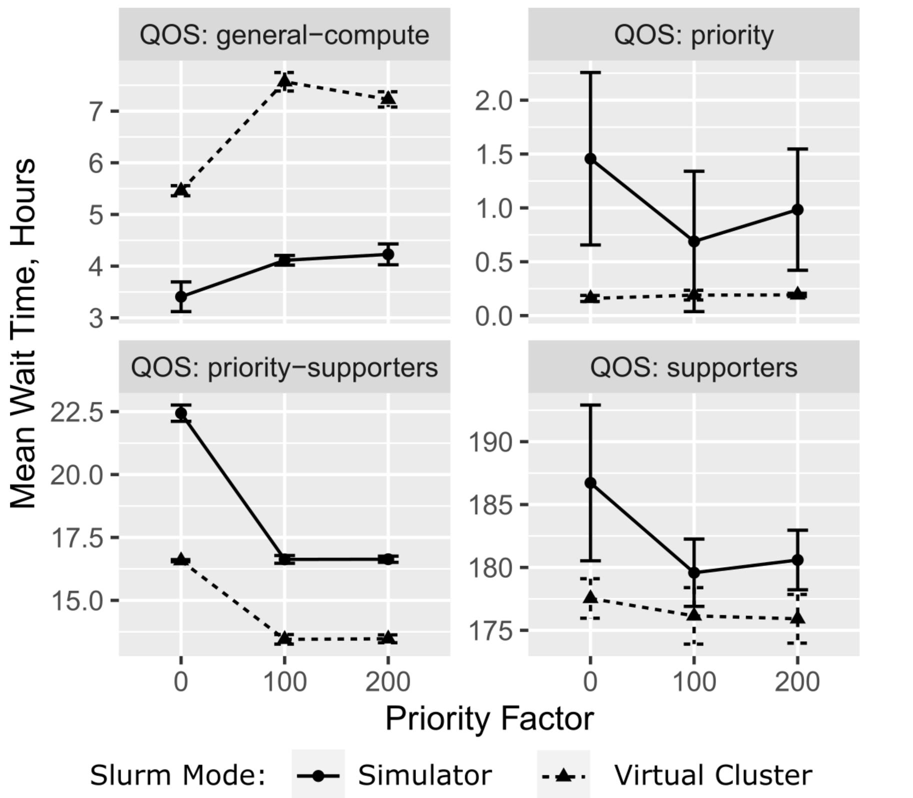
For micro-cluster both version of simulator (v2 and v3) are close to respective versions of Slurm.

Results: UB-HPC System, 217 nodes, 29,678 jobs



Distances between Real and Simulated workload realizations

New version of Slurm simulator produces results which are closer to actual Slurm than faster execution of previous version (new version is still faster).



To estimate the ability to predict relative values rather than absolute ones, we also vary the priority factor of several QoS groups (priority and supporters) while keeping the general QoS the same. For a Mid-sized system, we found that the absolute value of mean wait time differs between Virtual Clusters and Slurm Simulator. However, the trend is very similar, especially on higher values of mean wait time.

Conclusion

The new version of Slurm is sufficiently different from the older version, and thus, to assess the accuracy of a new simulator, we need to regenerate the reference workload realization (done with Virtual Clusters). So far, we have analyzed the accuracy on our small 10-node cluster, and the simulator shows results very similar to the actual Slurm. For the midsized system, we only compared the simulator to the older version of the actual Slurm, and the results are similar to our previous generation of Slurm simulator. We still need to further analyze the accuracy, however the initial conclusion is that we achieved a significant speed-up with little or no loss of accuracy. The resulting third version of our simulator has more than 500 times acceleration over real-time allowing simulation of a month-long workload in 90 minutes.

Acknowledgements

This work was supported by the National Science Foundation under awards OAC 2004954.
This work used compute resources at UB CCR and the ACCESS-CI through allocation CCR120014.

References

1. N.A. Simakov, R.L. DeLeon, Y. Lin, P.S. Hoffmann, and W.R. Mathias. 2022. Developing Accurate Slurm Simulator. In Practice and Experience in Advanced Research Computing (PEARC ’22), July 10–14, 2022, Boston, MA, USA. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3491418.3535178
2. N.A. Simakov, R.L. DeLeon, M.D. Innus, M.D. Jones, J.P. White, S.M. Gallo, A.K. Patra, and T.R. Furlani. 2018. Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC Systems by Utilization of Multiple Controllers and Node Sharing. In Proceedings of the Practice and Experience on Advanced Research Computing (Pittsburgh, PA, USA) (PEARC ’18). Association for Computing Machinery, New York, NY, USA Article 25, 8 pages. https://doi.org/10.1145/3219104.3219111
3. N.A. Simakov, M.D. Innus, M.D. Jones, R.L. DeLeon, J.P. White, S.M. Gallo, A.K. Patra, and Thomas R Furlani. 2017. A slurm simulator: Implementation and parametric analysis. In International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems. Springer International Publishing, Cham, 197–217. https://doi.org/10.1007/978-3-319-72971-8_10

Slurm Simulator code, Various utilities and documentation are available at

https://github.com/ubccr-slurm-simulator

Docker image in preparation