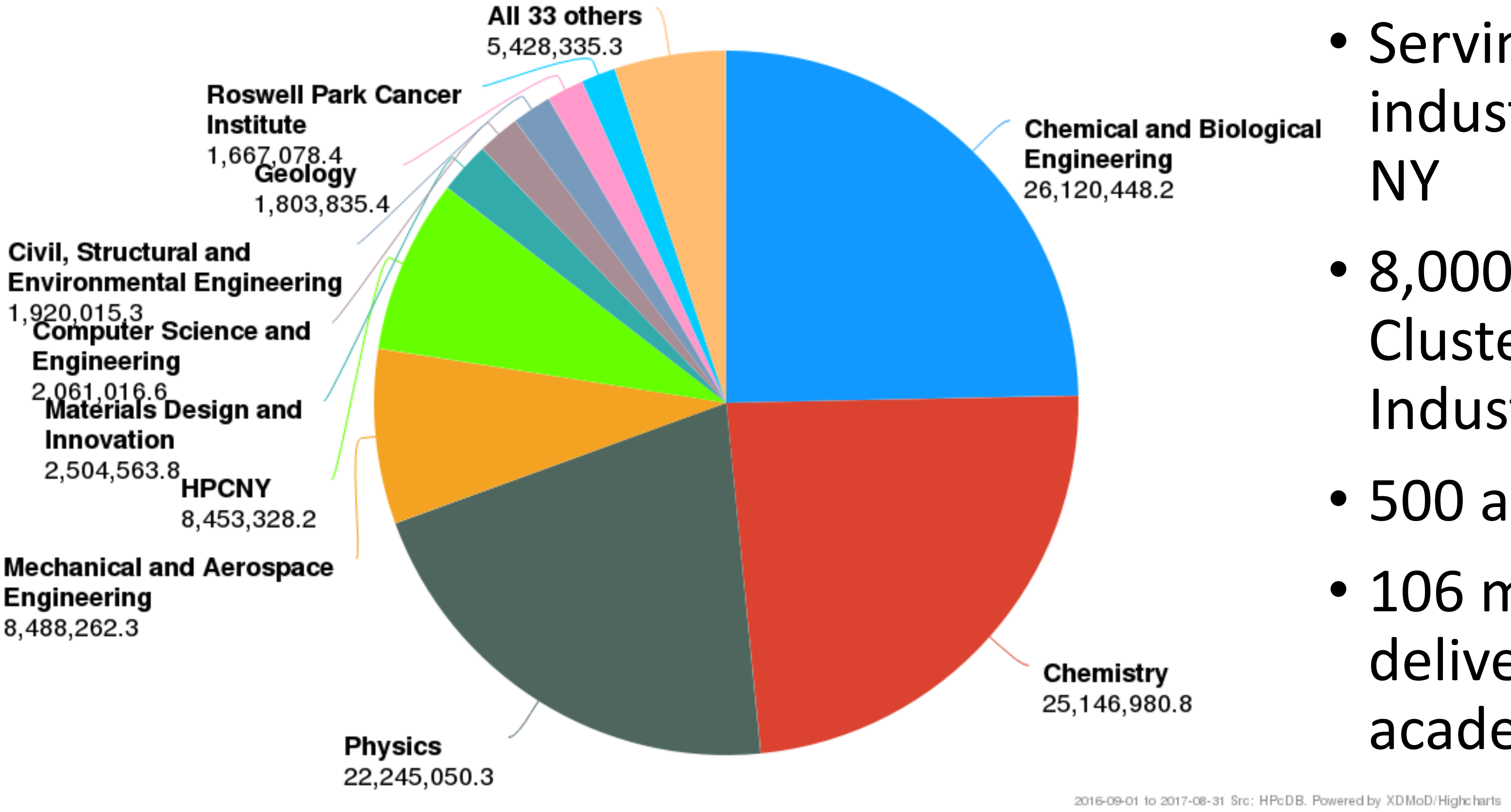
A Slurm Simulator: Implementation and Parametric Analysis

Nikolay A. Simakov, Martins D. Innus, Matthew D. Jones,Robert L. DeLeon, Joseph P. White, Steven M. Gallo, Abani K. Patra and Thomas R. Furlani

November 13, 2017, PMBS17 at SC17

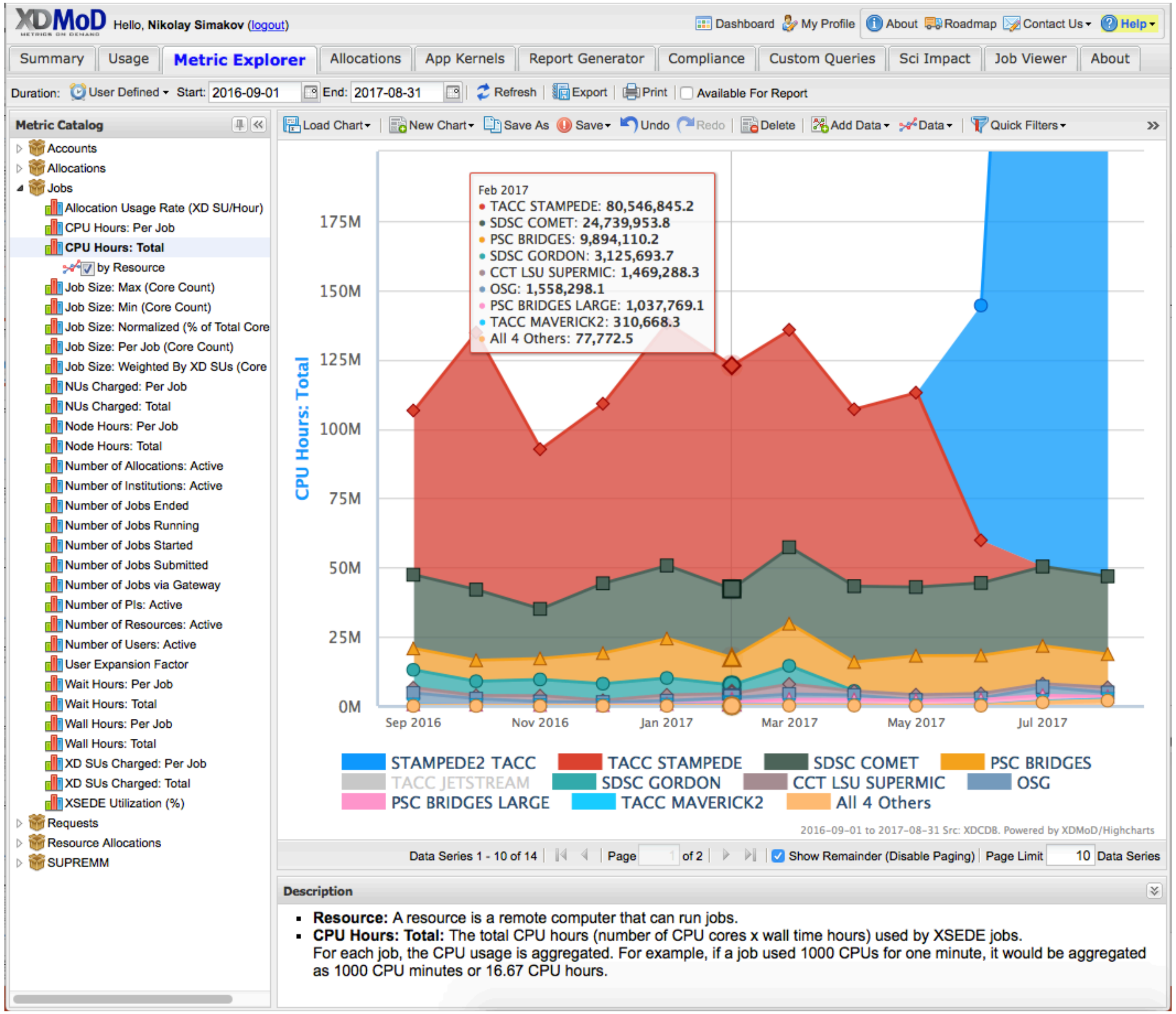
University at Buffalo
Center for Computational Research

Center for Computational Research, University at Buffalo



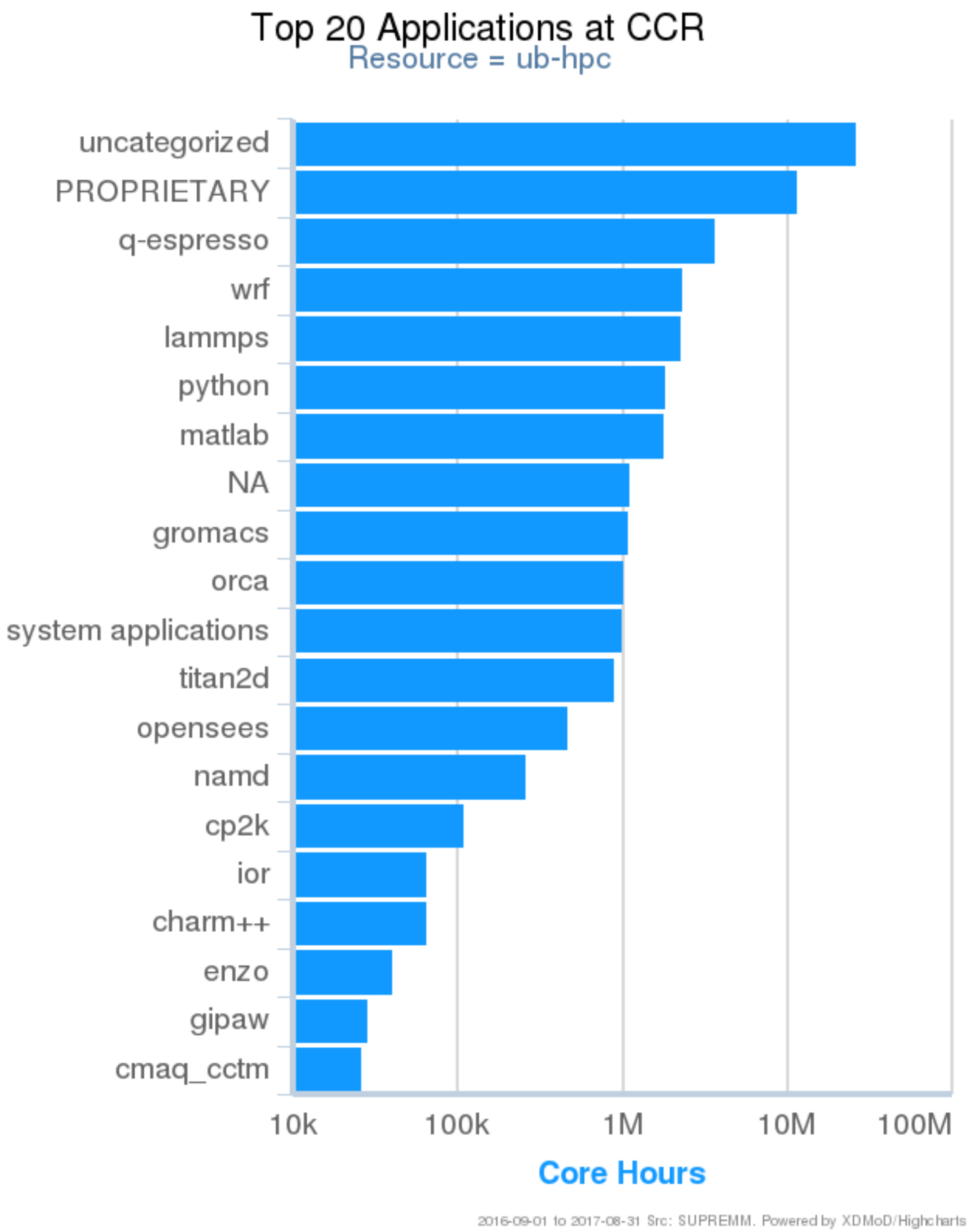
- Regional HPC Center
- Serving academic and industry users from western NY
- 8,000 Cores Academic Cluster, 3,456 Cores Industry Cluster
- 500 active users and 200 PI
- 106 millions cores hours delivered during 2016-2017 academic year

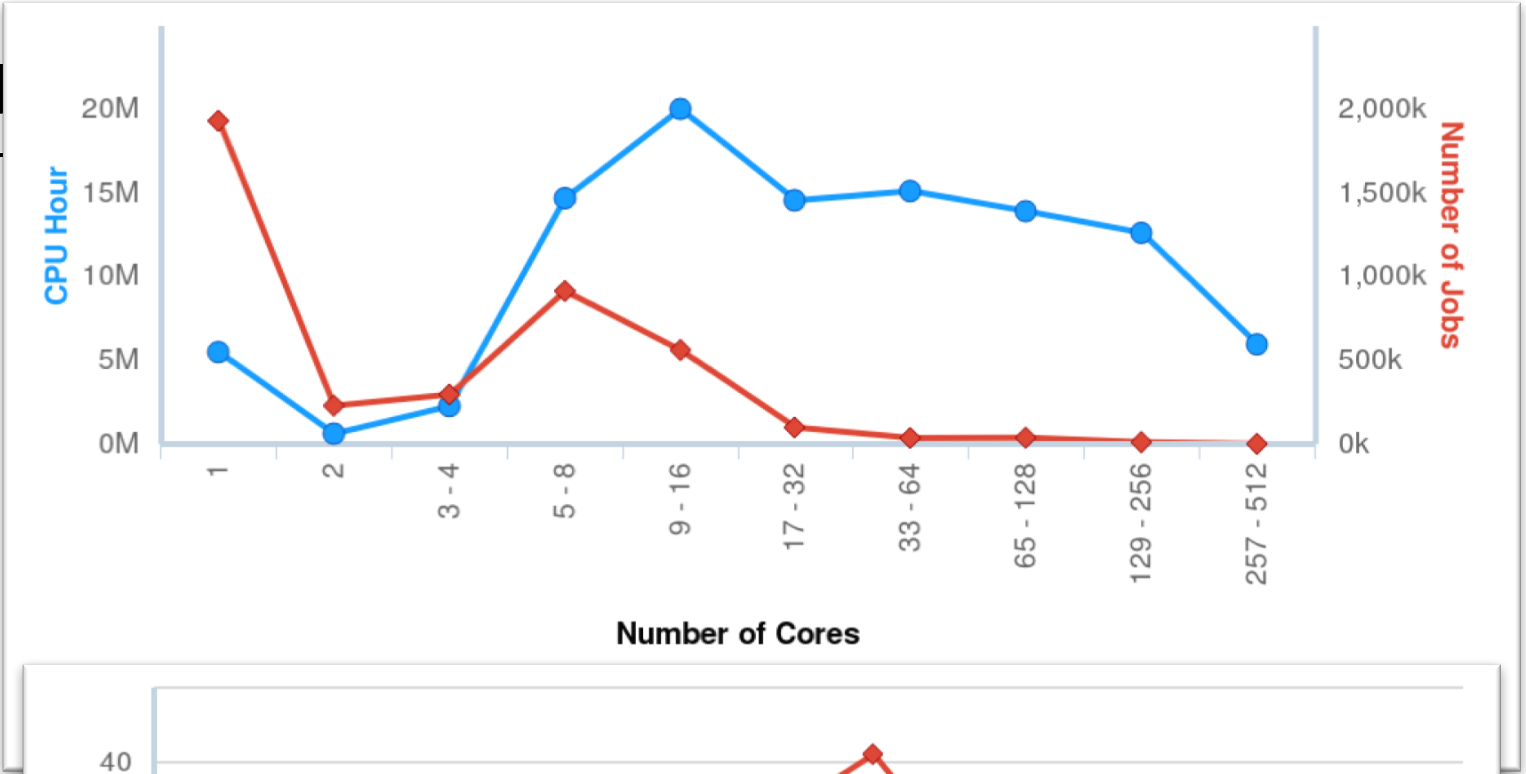
A Tool for HPC System Management

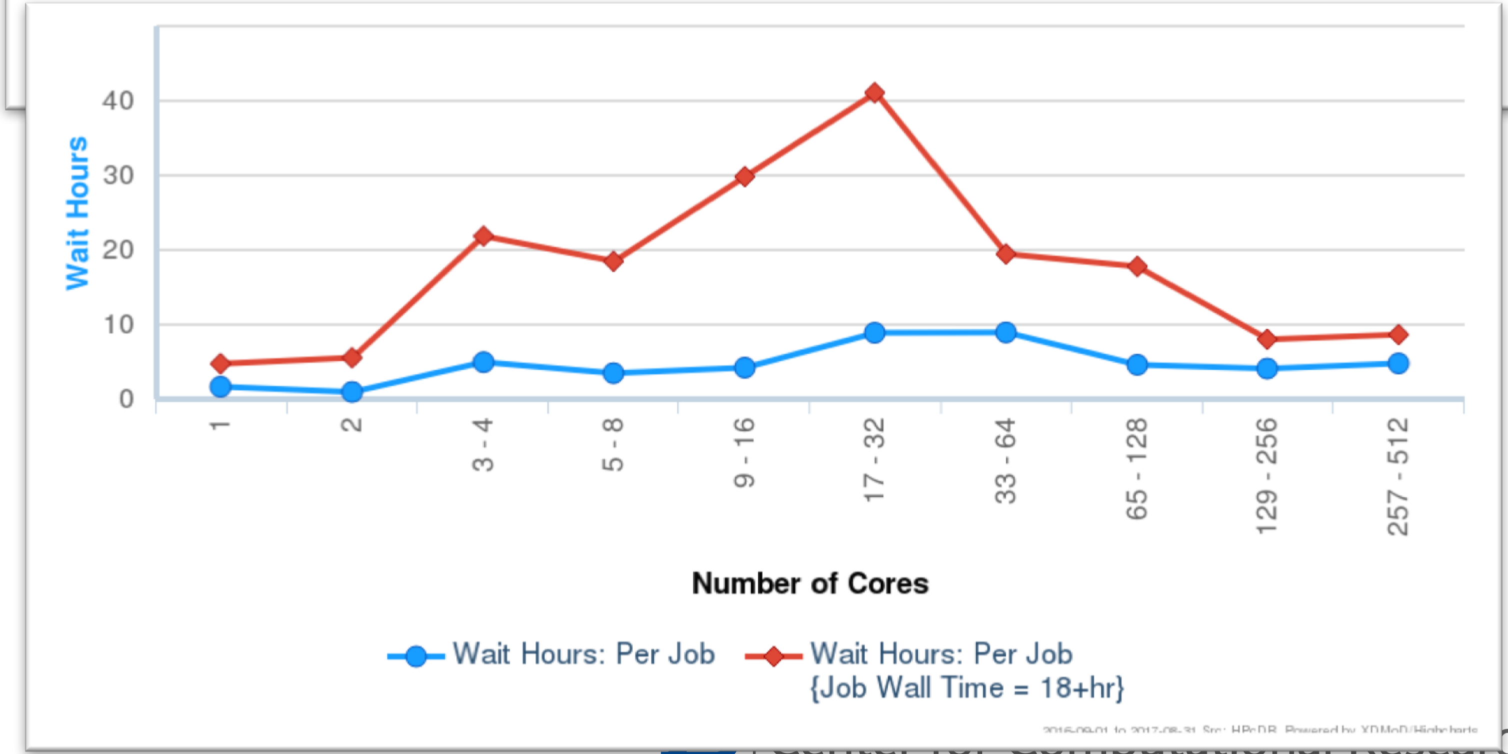


- **XDMoD: XD Metrics on Demand**
  - HPC resources usage and performance monitoring and analysis
  - On demand, responsive, access to job accounting data
  - NSF funded analytics framework developed for XSEDE
  - **http://xdmod.ccr.buffalo.edu/**
- **Comprehensive Framework for HPC Management**
  - Support for several resource managers (Slurm, PBS, LSF, SGE)
  - Utilization metrics across multiple dimensions
  - Measure QoS of HPC Infrastructure (App Kernels)
  - Job-level performance data
- **Open XDMoD*: Open Source version for HPC Centers**
  - 100+ academic & industrial installations worldwide
  - http://open.xdmod.org/
- **Utilized for Blue Waters Workload Analysis**
- **Currently Carrying out "XSEDE" Workload Analysis**

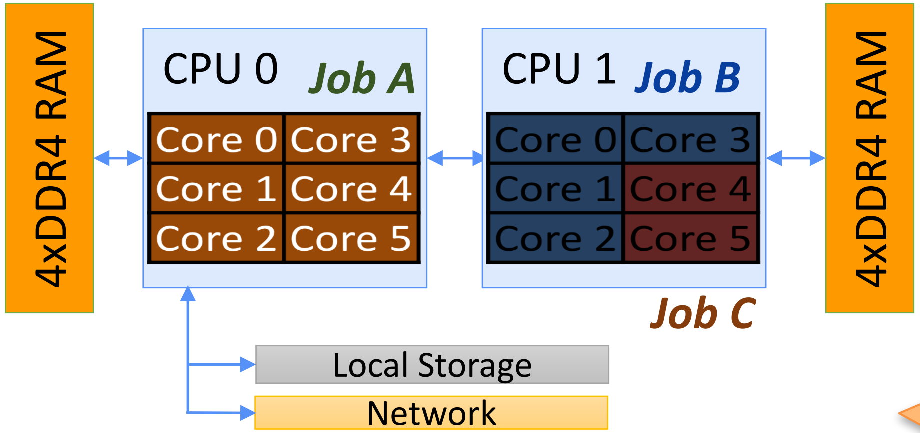
Center for Computation







Effect of Node Sharing



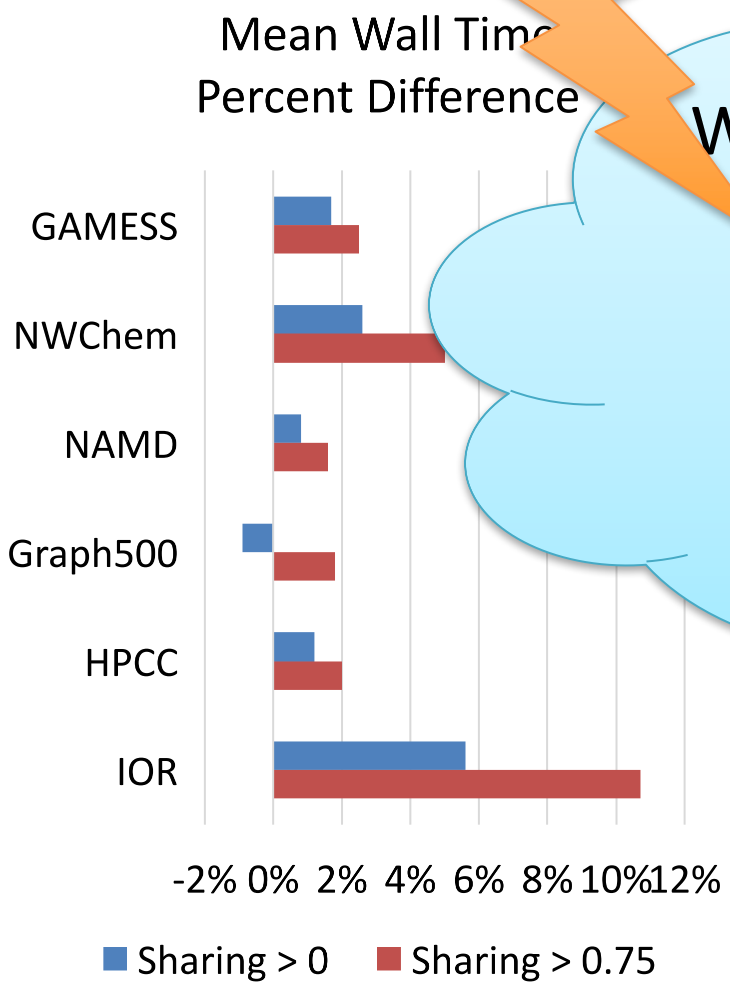
- Under node sharing several jobs are allowed to be executed on same node.
- Jobs has dedicated cores.

- There are computational tasks which cannot efficiently use all of the cores available on a node
  - serial applications
  - poorly scalable parallel software
  - small problem sizes
  - Time imbalanced embarrassingly parallel tasks such as parameter sweeps

But how does it affect the application performance?

Effect of Node Sharing

Single Core



Single Socket

- Th… effect of node … on application … is small.

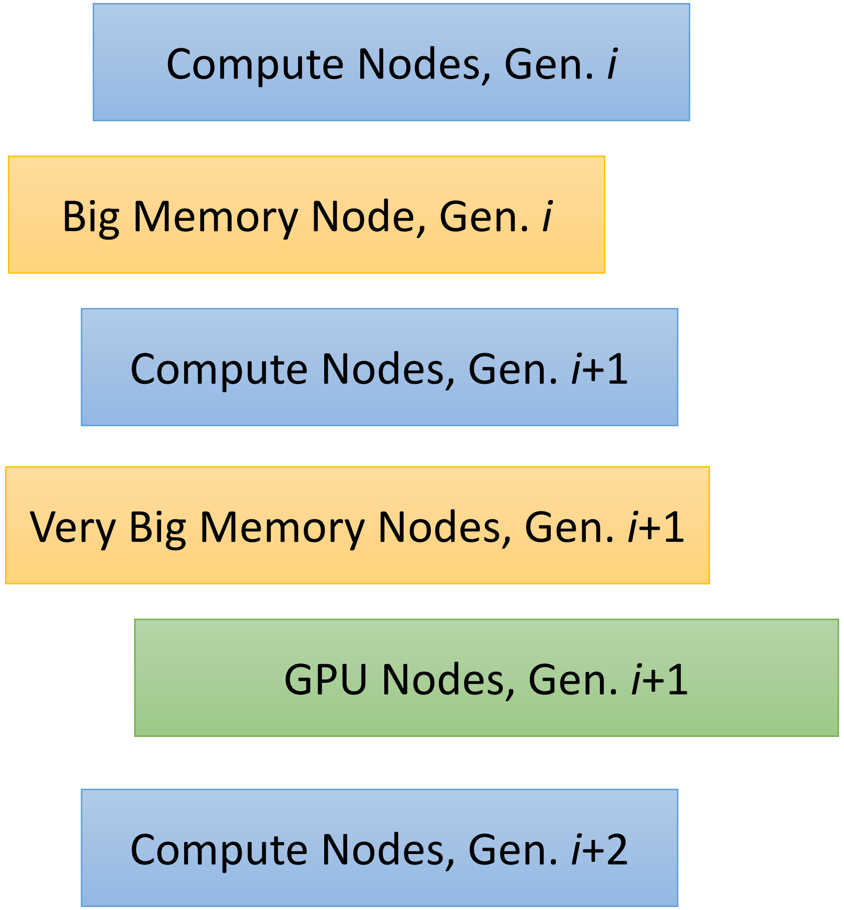
What is the overall effect on the whole system?

Will it actually increase throughput?

To have quantitative answer we need workflow simulator!

Simakov et al., 2016. *Proceedings of the XSEDE16.*

Slurm Workload Manager



- Slurm is an open-source resource manager for HPC
- It provides high configurability for inhomogeneous resources and job scheduling
- It is used on large range of HPC resources from small to very large systems.

- Which configuration is best for particular needs?

Slurm Simulator

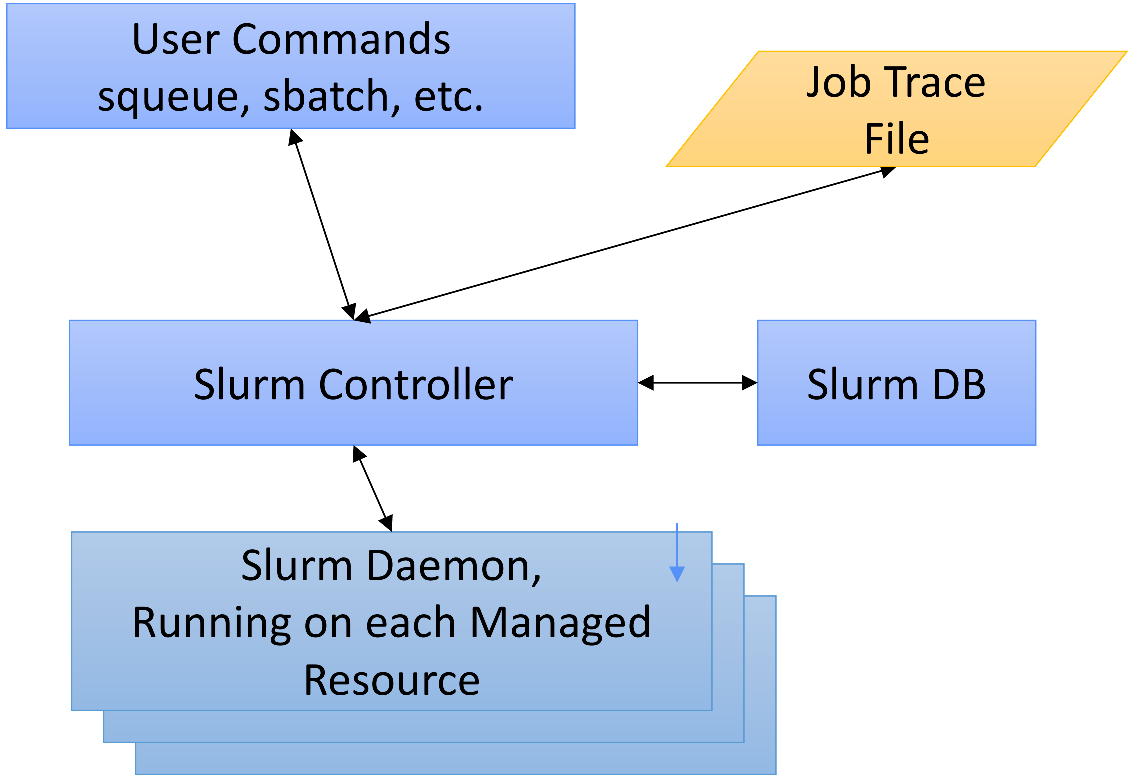
- Why do We Need Slurm Simulator?
  - To check Slurm configuration prior it deployment
  - Finding most optimal parameters for Slurm
  - Modeling of future systems

- Workflow Simulators:
  - Bricks, SimGrid, Simbatch, GridSim and Alea, Maui and Moab Scheduler

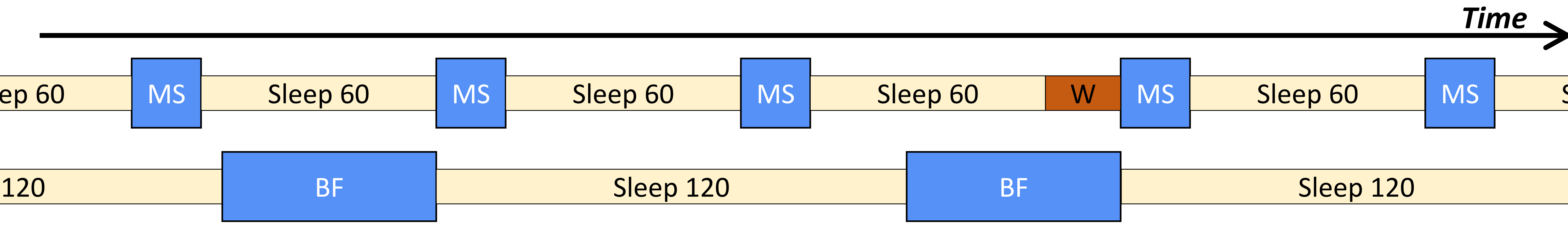
- Slurm Simulators:
  - Original version developed by Alejandro Lucero
  - Later improved by Trofinoff and Benini

  - Works only for very small systems and Slurm Version is outdates.

Making Slurm Simulator from Slurm

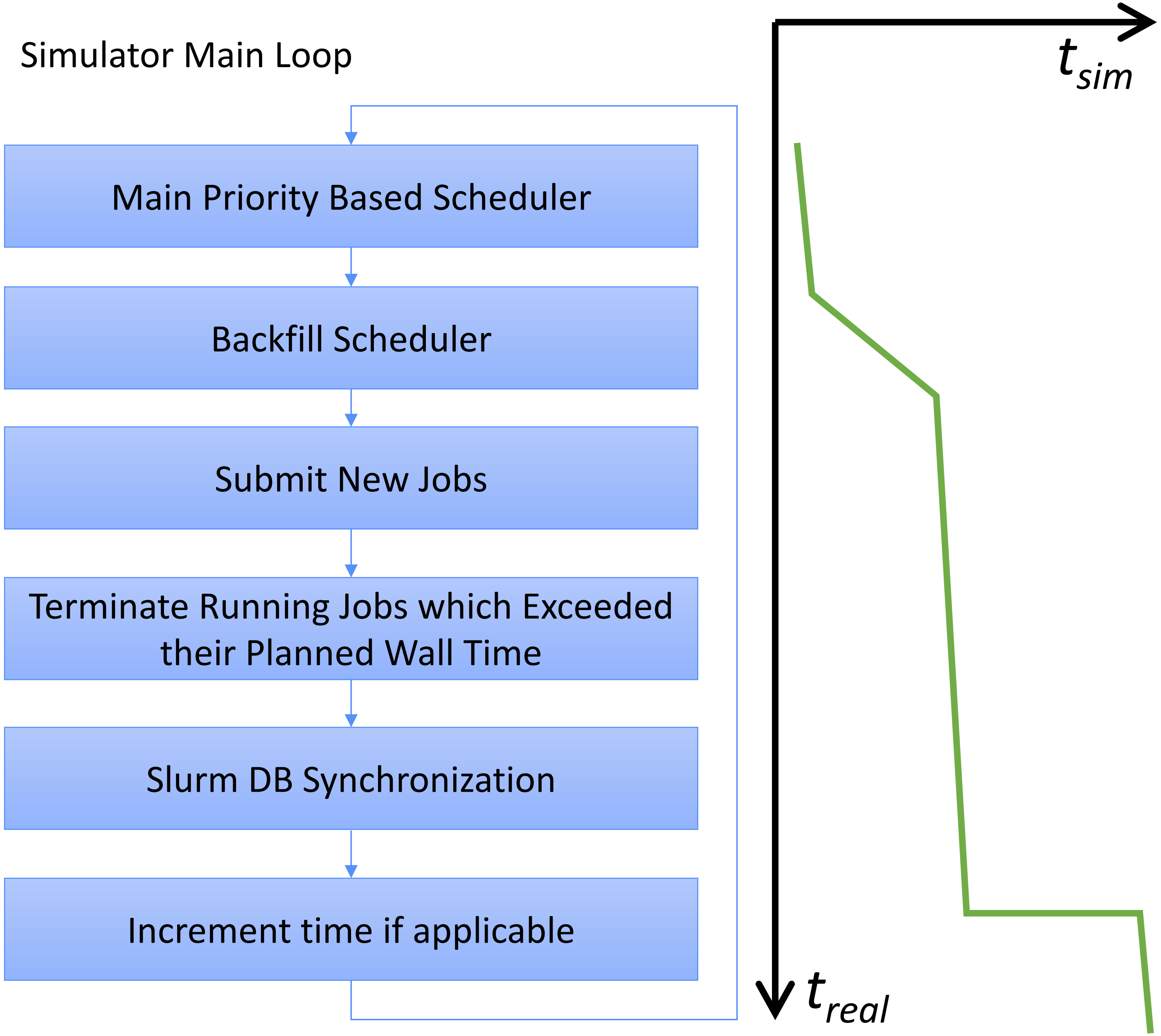


- Started from latest stable Slurm Release
- Minimized number of processes
- Slurm Controller performs simulation
- Serialize Slurm Controller



Serialization of Slurm Controller

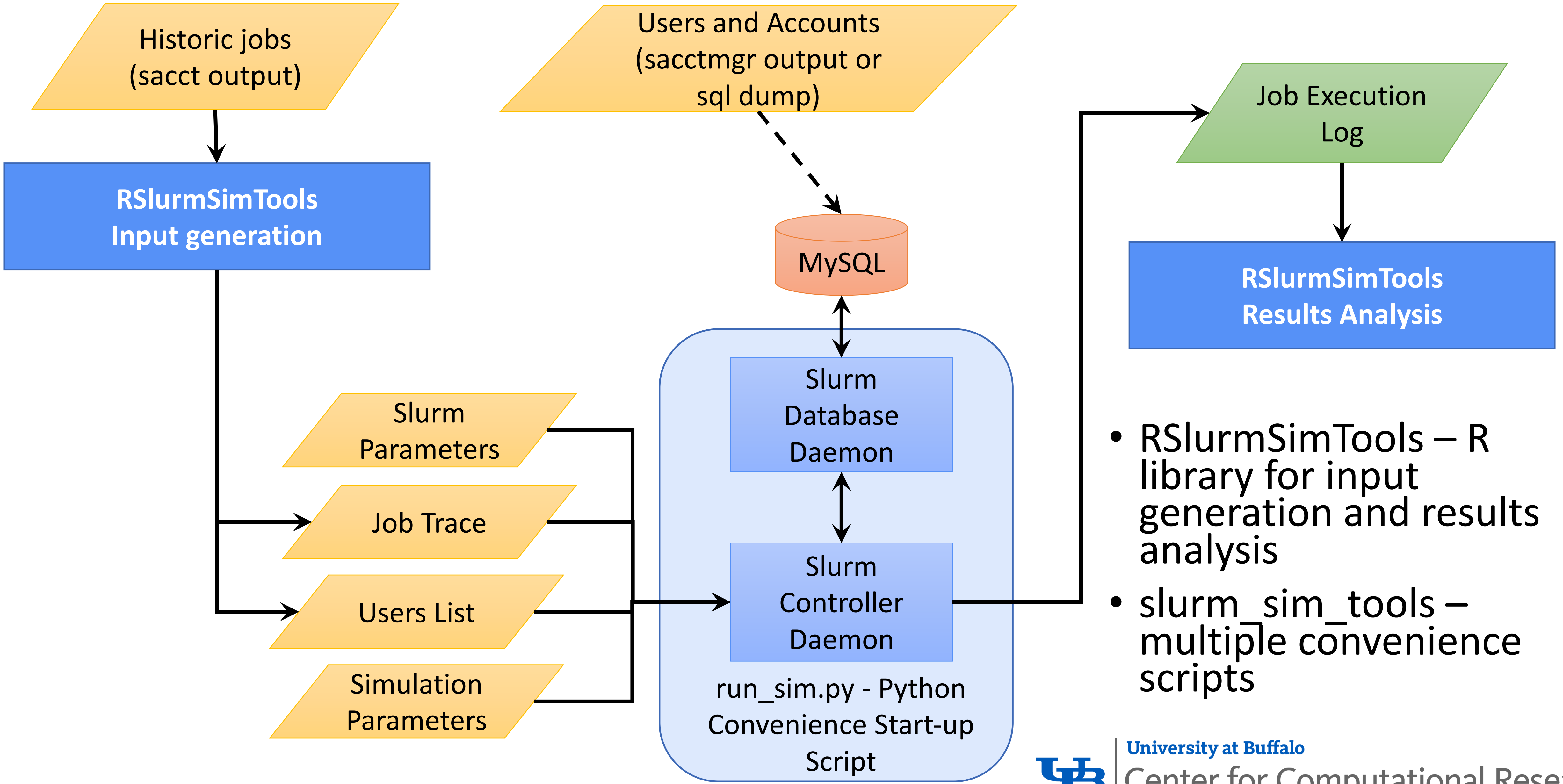
Simulator Main Loop



- Serialized Slurm Controller
  - In real Slurm scheduling is efficiently serial due to thread locks
  - Due to lacking of threads locks, compilation with optimization flags on and with assert functions off, the backfill scheduler is about **10 times faster in simulation mode**

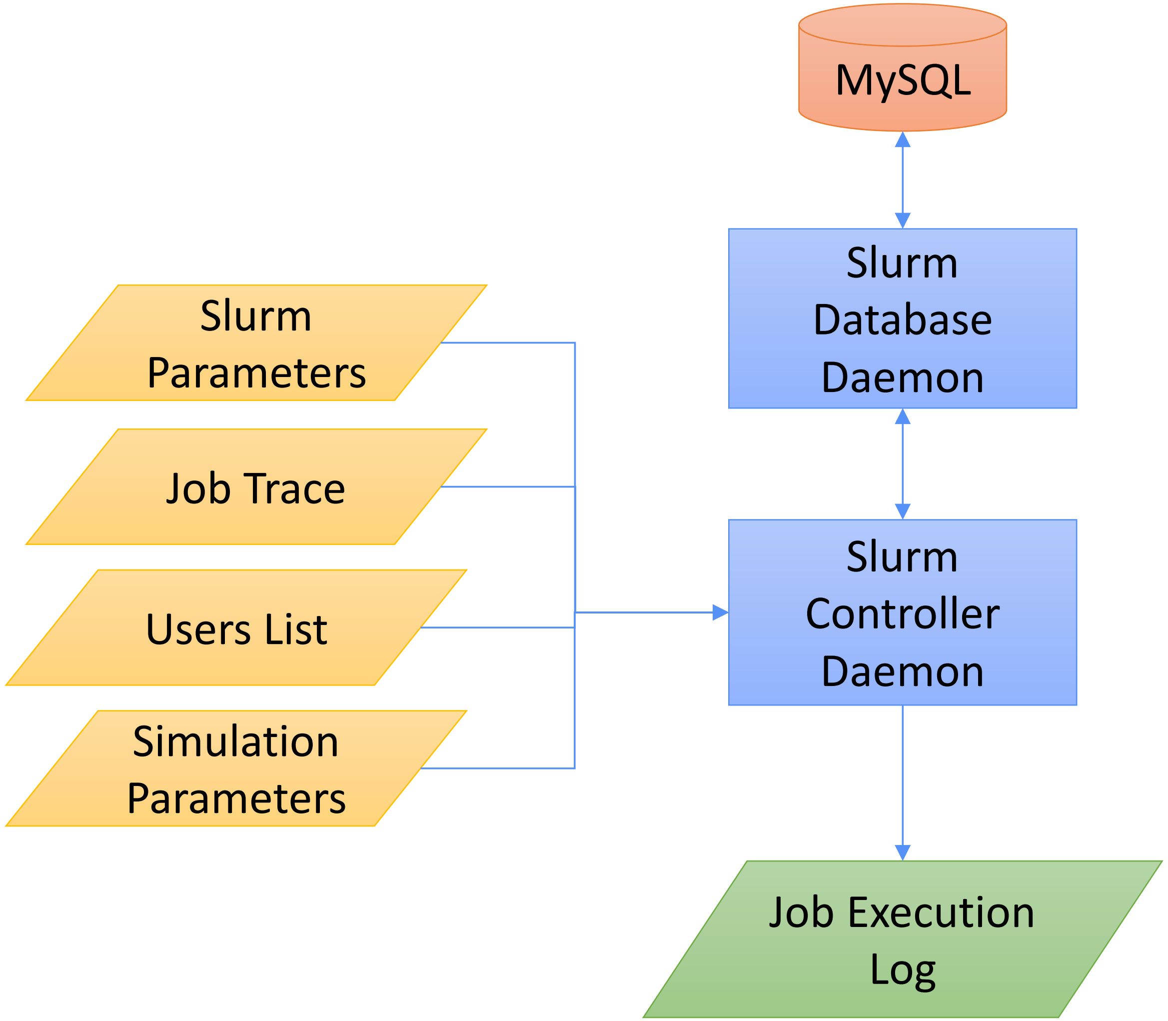
- Simulating time –Scaled real-time with time stepping
  - Shifted real time in most places
  - Shifted and scaled real time in backfill scheduler
  - Time increment (30-60 seconds) in case of no events

Simulating Workflows with Slurm Simulator



- RSlurmSimTools – R library for input generation and results analysis
- slurm_sim_tools – multiple convenience scripts

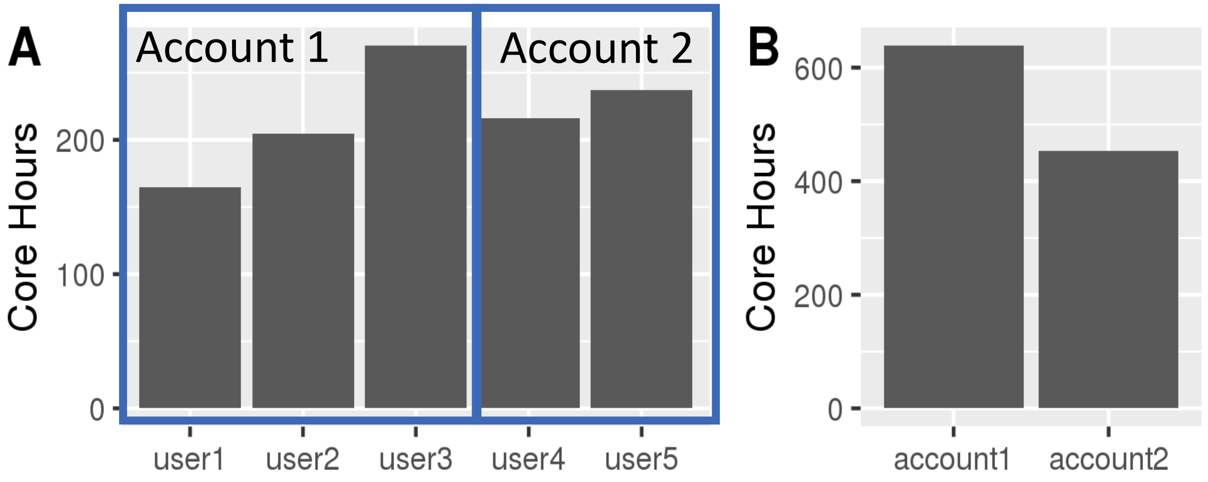
Slurm Simulator: Implementation Details



- For better performance number of process and thread was decreased
- Communication with Slurmd mimicked within Slurmctld
- Slurmctld was serialized, function are executed serially from main simulation event loop
- Slurm compiled with optimization flags and with assert functions off
- Time is scaled within backfill scheduler usually by factor of 10 to reflect difference in performance of real Slurm and Slurm simulator
- At the end of main simulator event loop time is incremented by 1 seconds if no event happened during the loop
- R-scripts is used to generate job trace files
- R-scripts is used to analyzed results

Validating Simulator using, Micro-Cluster, Small Model System

Node Type	Number of Nodes	Cores per Node	CPU Type	RAM
Compute	4	12	CPU-N	48GB
Compute	4	12	CPU-M	48GB
High Memory	1	12	CPU-G	512GB
GPU Compute	1	12	CPU-G	48GB

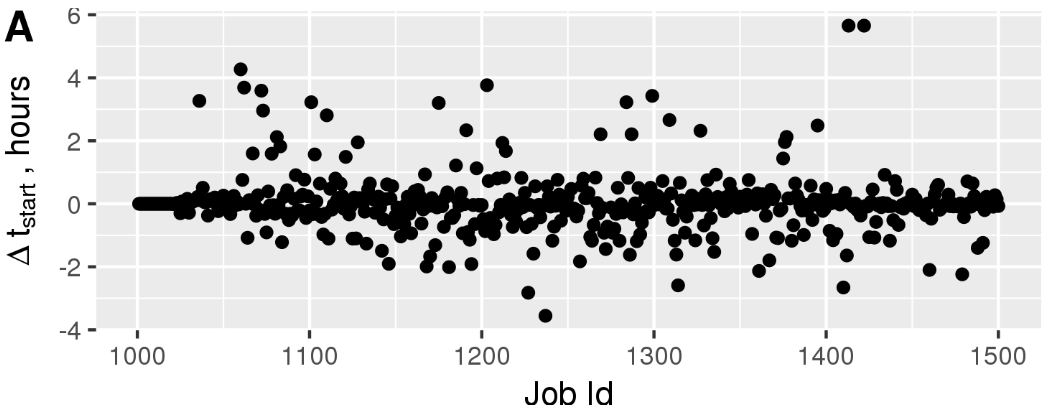


- Micro-Cluster is small model cluster created for Slurm simulator validation
- Reference data was obtained by running regular Slurm in front-end mode
- Micro-Cluster configuration was chosen to test constrains, GRes, cores and memory as consumable resources
- The workload consisted of 500 jobs and takes 12.9 hours to complete
- 5 users belonging to 2 accounts

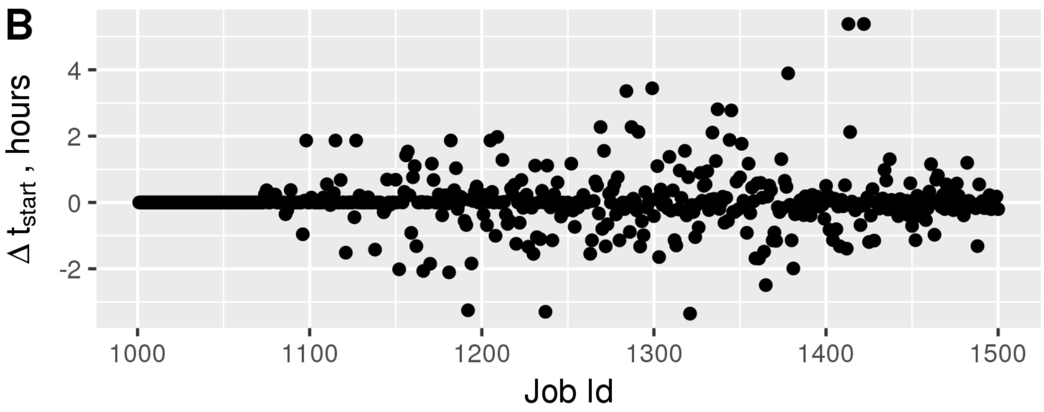
Micro-Cluster: Slurm Scheduling is not Unique

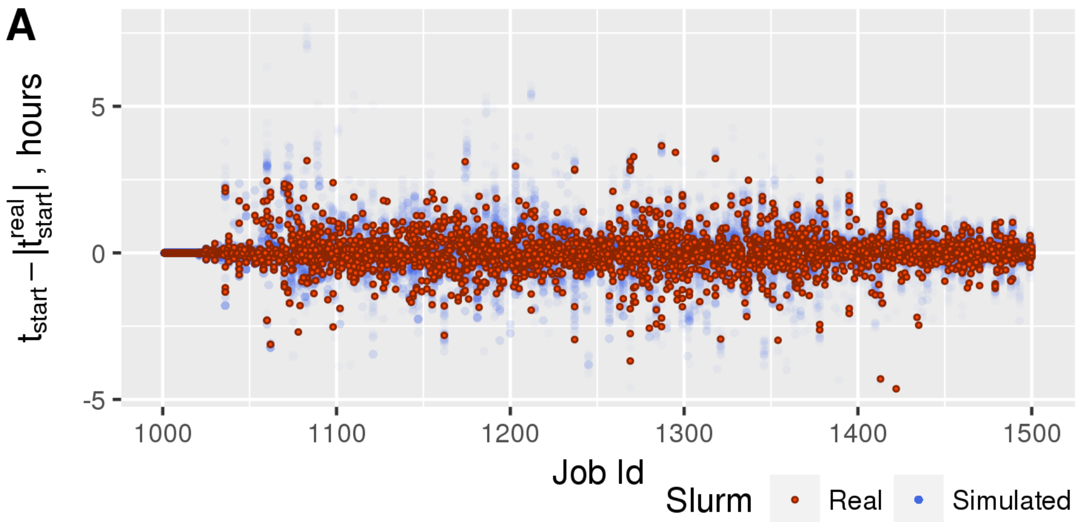
Job start time difference:

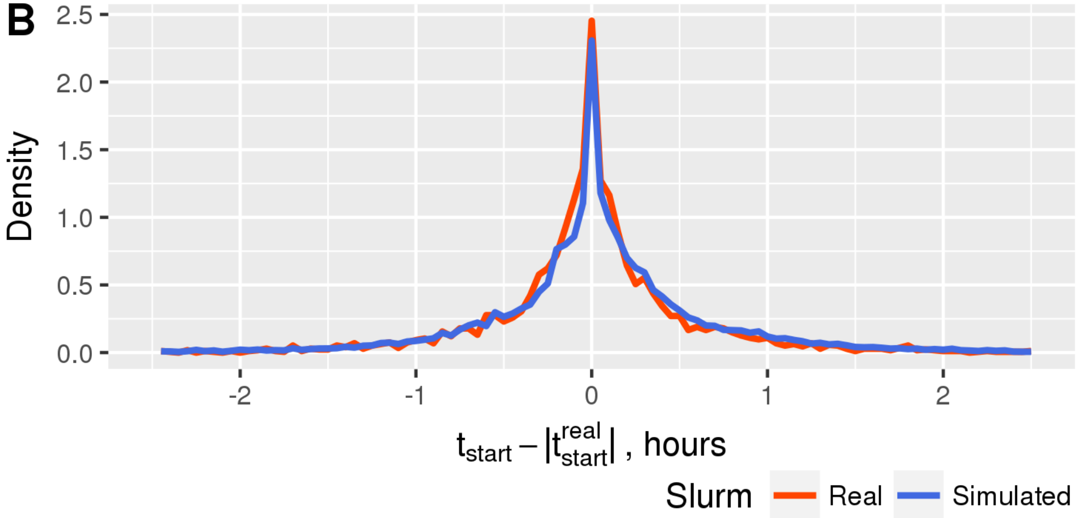
Between simulated and real Slurm runs



Between two real Slurm runs.



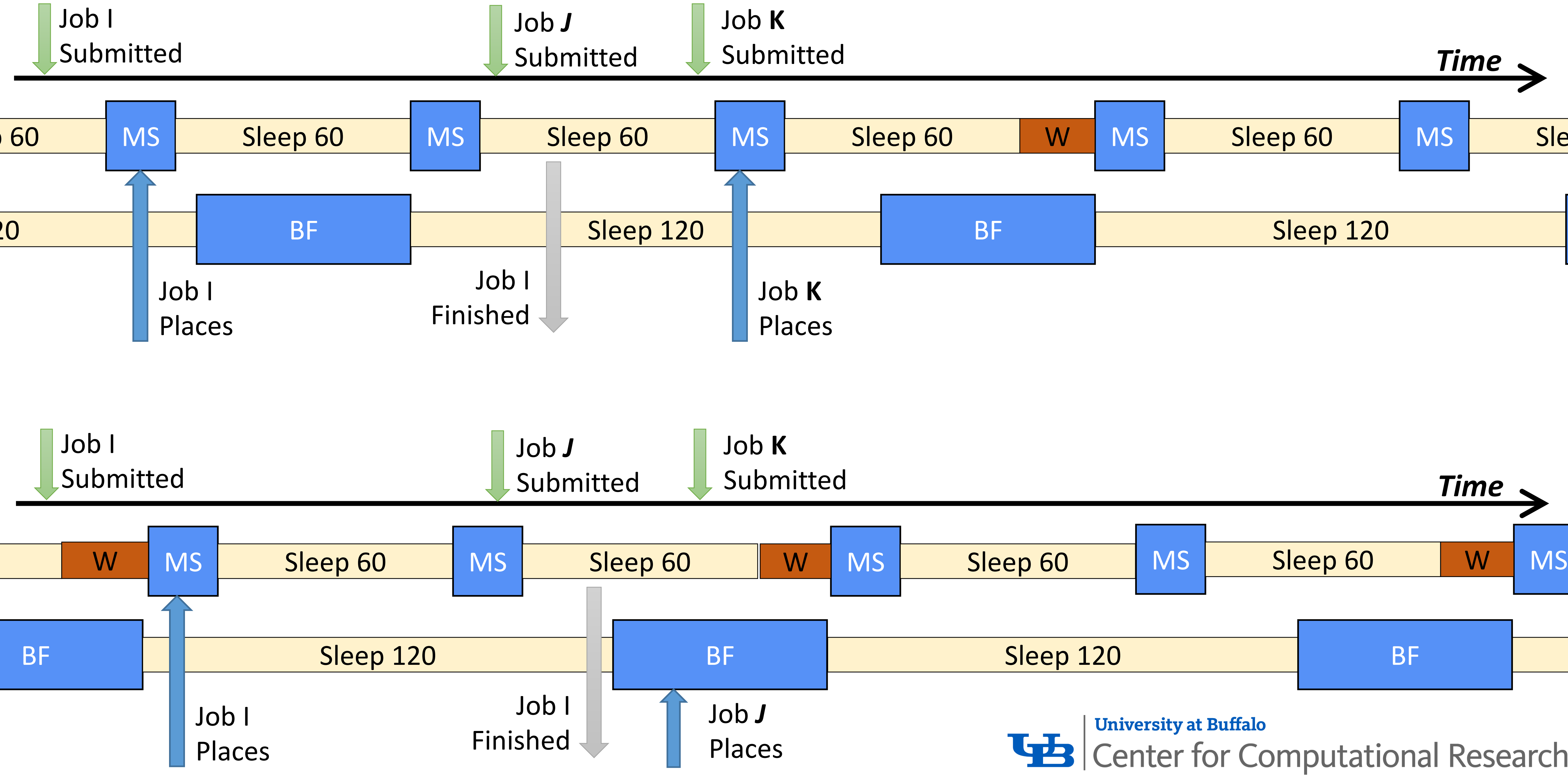




- Simulation has similar variability to real Slurm

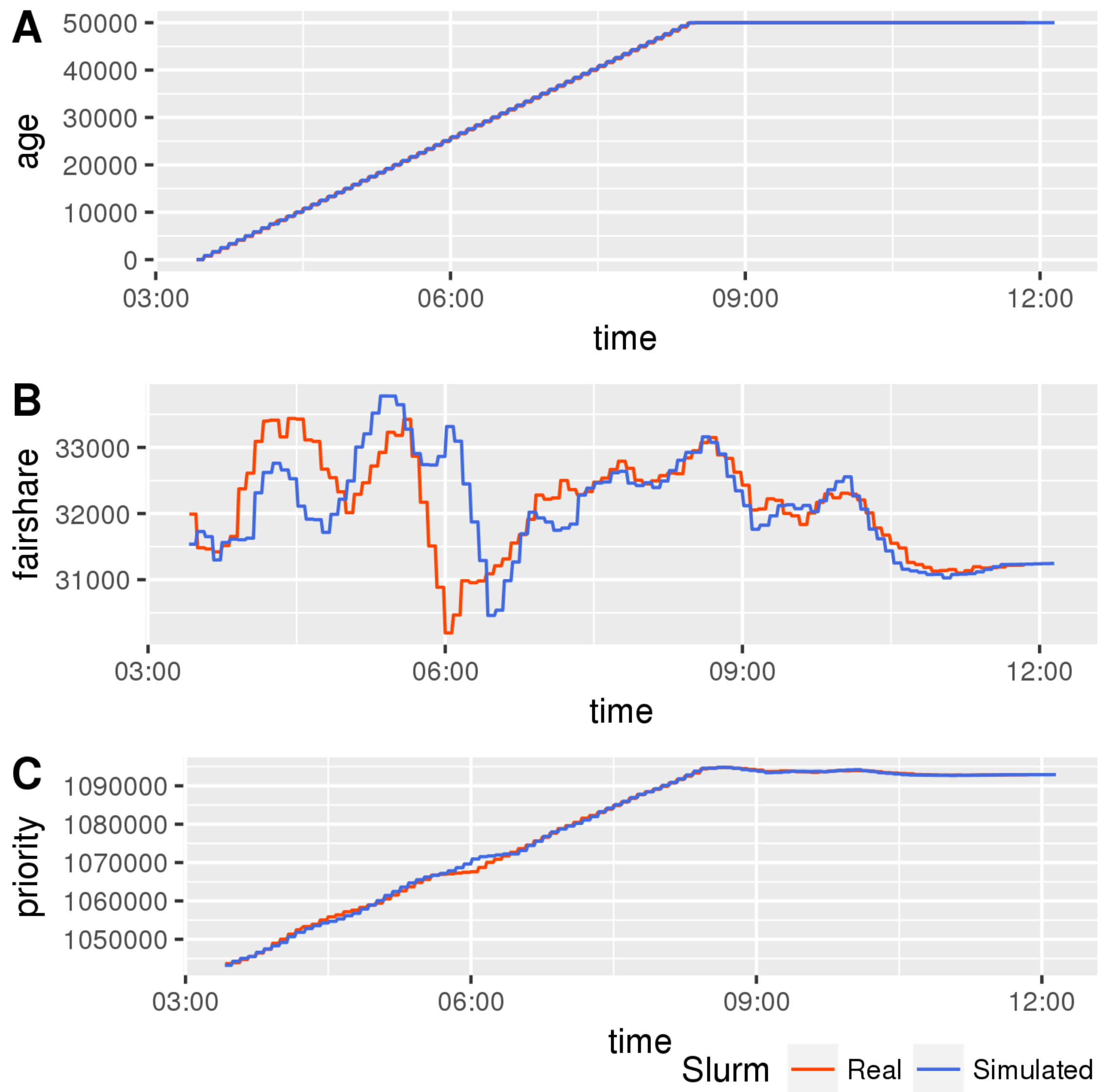
Variability Origin

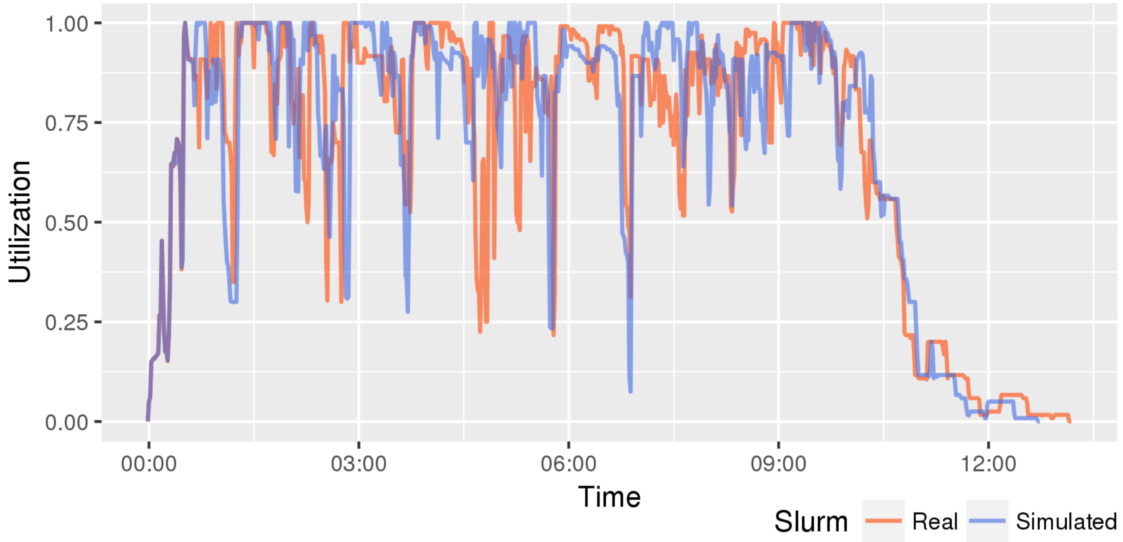
Job J has lower priority then Job **K**



Micro-Cluster: Comparison of Utilization and Job Priorities

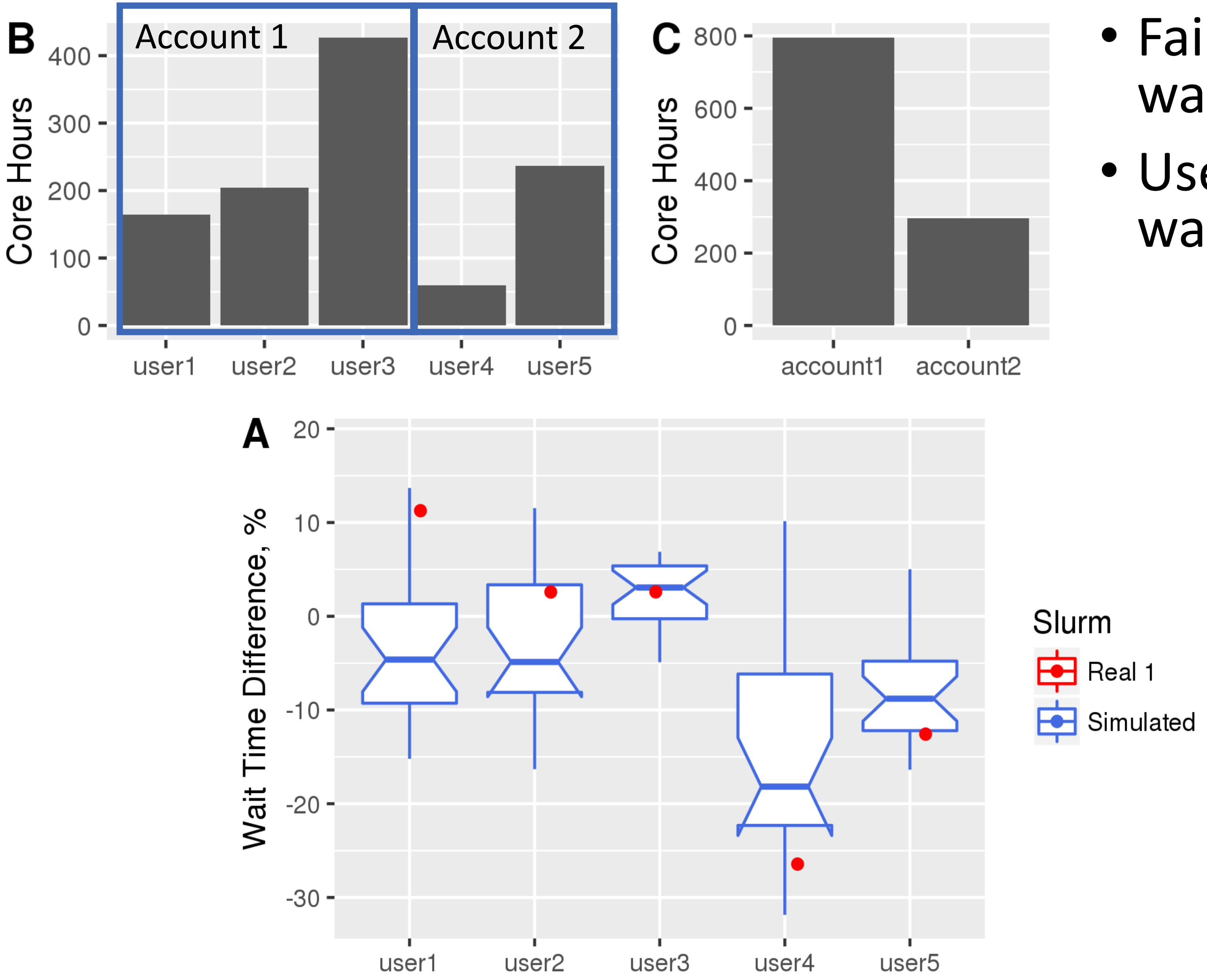
Changing of Job Priority Factor over Time





- Resource utilization and job priority changes in simulation is similar to real Slurm

Micro-Cluster: Modifying fair-share priority factor weight



- Fair-share priority factor weight was increased by 20%
- User 1 should biggest decrease in wait time

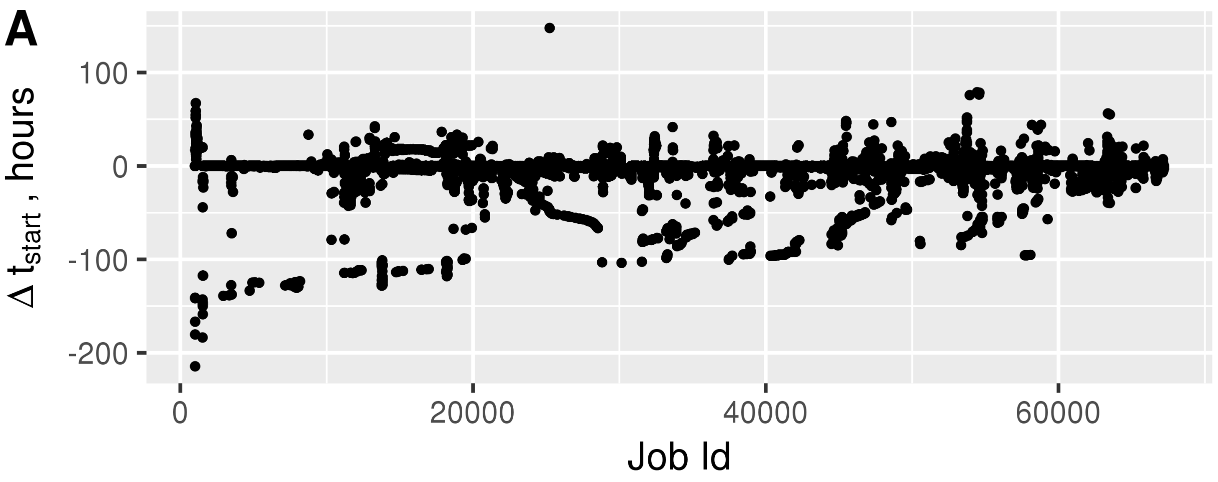
- Real wait time are within the range of predicted values

Studying UB-HPC Cluster

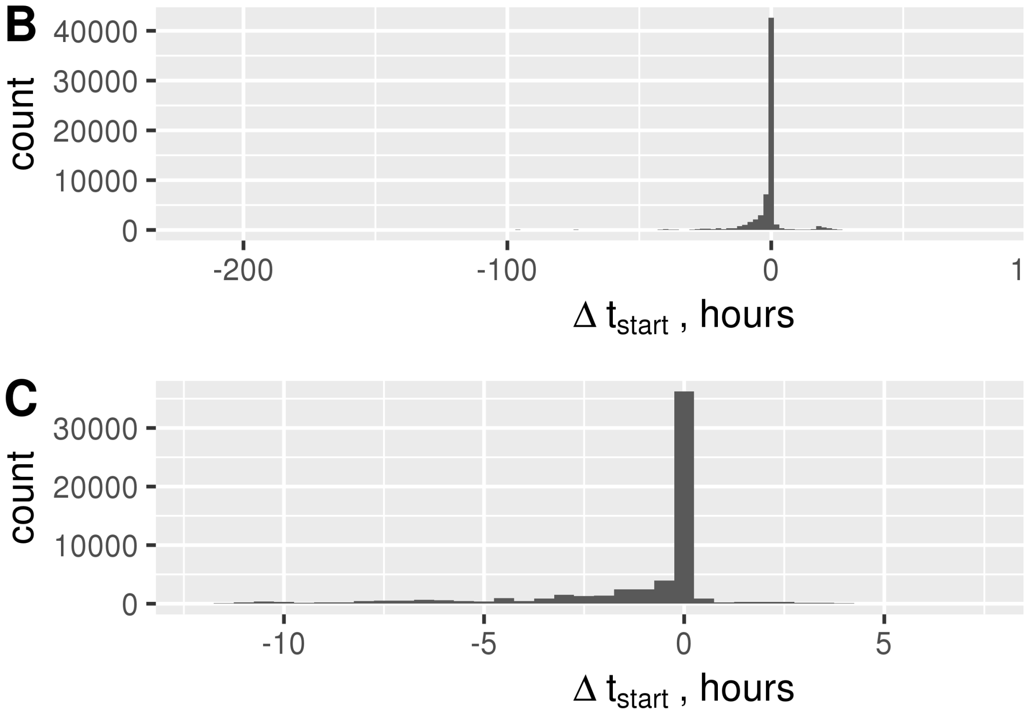
Node Type	Number of Nodes	Cores per Node	CPU Type	RAM
Compute	32	16	Intel E5-2660	128GB
Compute	372	12	Intel E5645	48GB
Compute	128	8	Intel L5630	24GB
Compute	128	8	Intel L5520	24GB
High Memory	8	32	Intel E7-4830	256GB
High Memory	8	32	AMD 6132HE	256GB
High Memory	2	32	Intel E7-4830	512GB
GPU Compute	26	12	Intel X5650	48GB

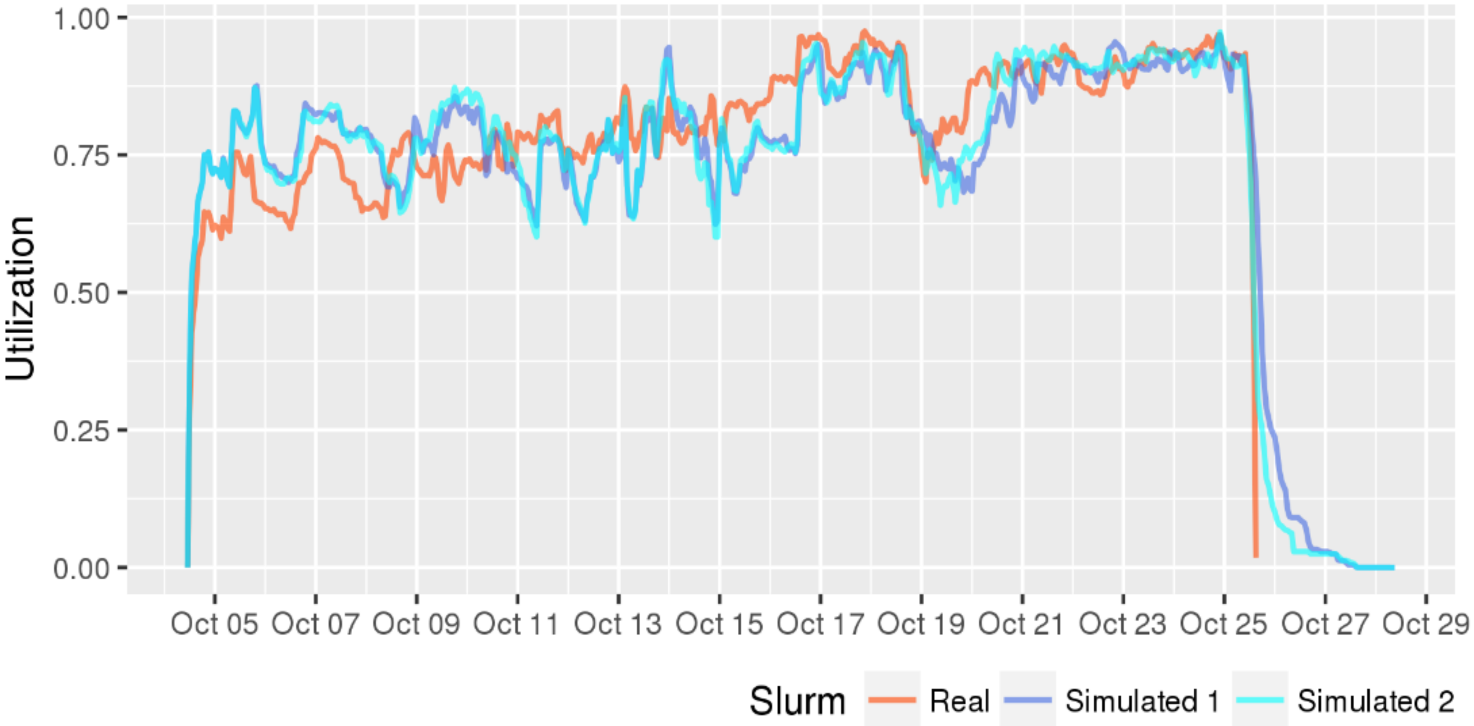
Historic workload for 24 days was used (October 4, 2016 to October 28, 2016)

Studying UB-HPC Cluster: Simulation vs Historic Data

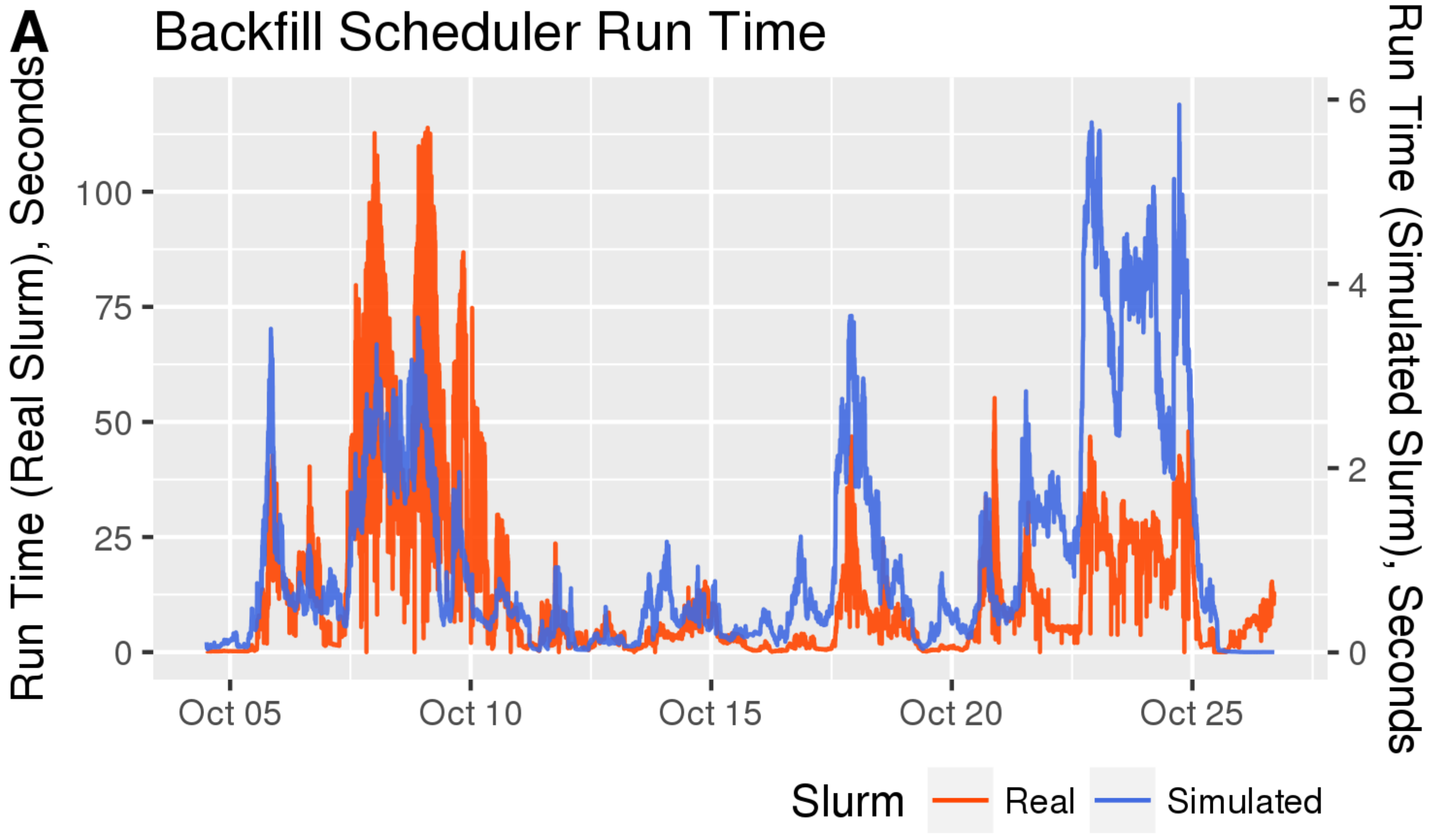


- Simulation was not having initial historic usage therefore initial fair-share priorities were incorrect





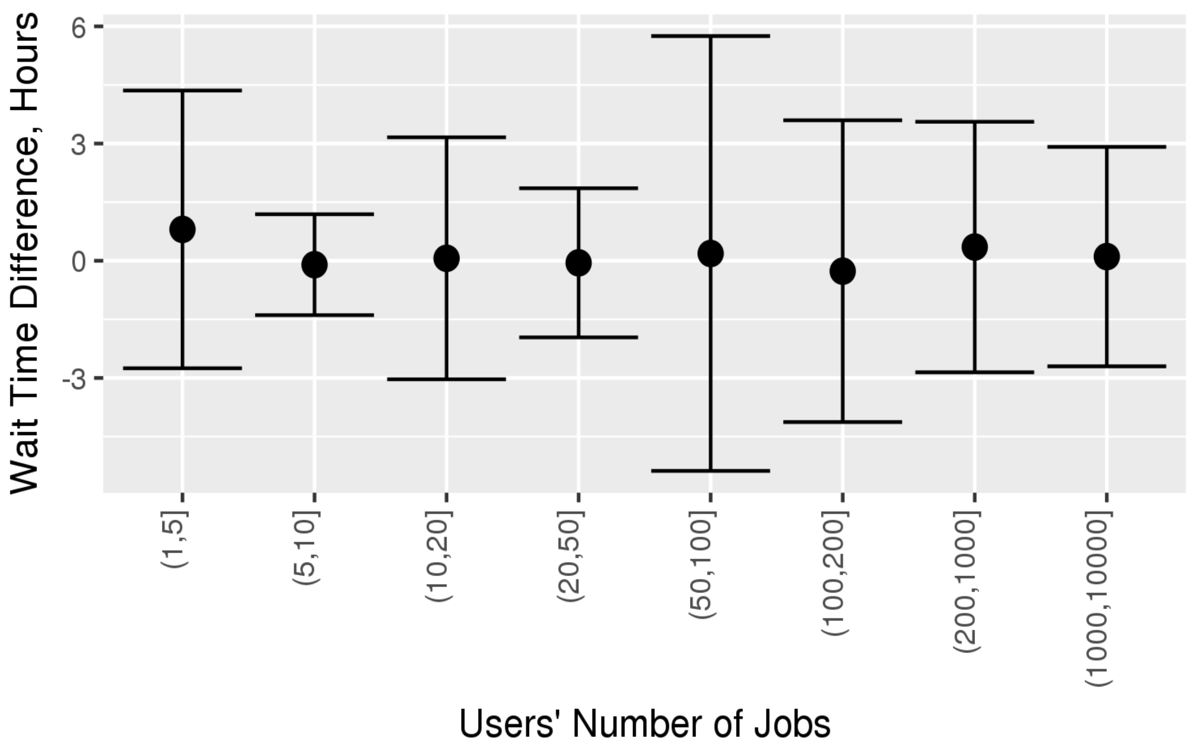
Studying UB-HPC Cluster: Simulation vs Historic Data



- Missing the influence from excessive RPC calls, the performance hit from multiple threads started for jobs start-up and finalization

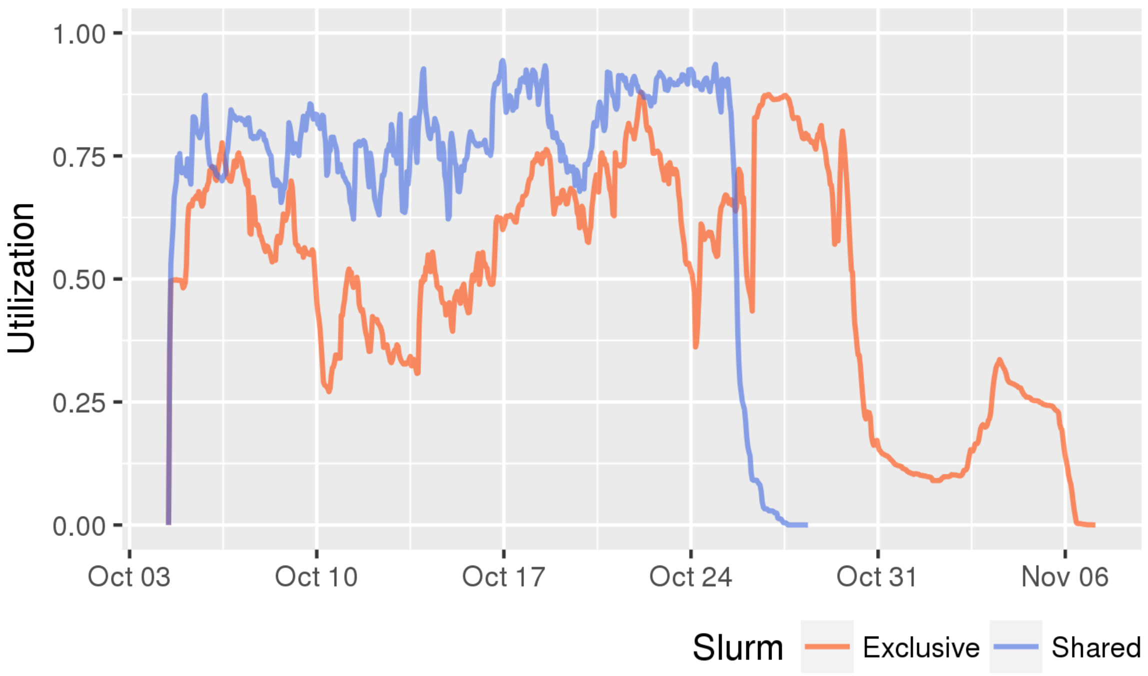
Reducing *bf_max_job_user*

- *bf_max_job_user* specifies maximal number of user's jobs considered by backfill scheduler for scheduling
- *bf_max_job_user* was reduced from 20 to 10



- 0.1% (40 minutes or 0.1%) increase in time to complete the workload
- The mean wait time is 8 minutes longer and the standard deviation of the wait time differences is 3 hours.
- 25% decrease in the number of jobs considered for scheduling
- 30% decrease in backfill scheduler run time.

UB-HPC Cluster: Node sharing



- The exclusive mode takes 10.8 more days (45% more time) to complete the same workload
- The average increase in waiting time is 5.1 days with a standard deviation of 6.6 days.
- The 45% increase in time to complete the same load can be translated into the need to have a 45% larger cluster to serve the same workload.

Studying Stampede 2

Node Type	Number of Nodes	Cores per Node	CPU Type	RAM
Intel Knights Landing	6400	68	Intel Xeon Phi 7250	96GB+16GB
Intel Xeon Skylake-X	1736	48	Intel Xeon Platinum 8160	192GB

- Node Sharing on Skylake-X Nodes
  - Sharing by Sockets or by Cores
- Separate Controller For KNL and Skylake-X Nodes

- 12 weeks workload was generated from stampede 1 historic workload (2015-05-16 to 2015-08-08)
- Number of jobs was scaled proportional to node count
- Sub-node jobs was calculated using CPU utilization

Stampede 2. Node Sharing and Separate Slurm Controllers

Controller	Node Sharing on SKX Nodes	Wait Hours, Mean	Wait Hours, Mean Weighted by Node Hours
Jobs on SKX Nodes			
Single	no sharing	**10.9 (0%)**	17.0 (0%)
	sharing by sockets	8.2 (-25%)	15.5 (-9%)
	sharing by cores	8.2 (-24%)	15.5 (-9%)
Separate	no sharing	**7.1 (-35%)**	15.0 (-12%)
	sharing by sockets	5.3 (-51%)	13.8 (-19%)
	sharing by cores	**5.5 (-49%)**	13.9 (-18%)
Jobs on KNL Nodes			
Single	no sharing	8.6 (0%)	9.2 (0%)
	sharing by sockets	7.2 (-16%)	9.2 (-1%)
	sharing by cores	7.3 (-15%)	9.1 (-1%)
Separate	no sharing	8.2 (-4%)	9.4 (2%)

- Node sharing and separate controllers cut waiting times nearly in half

Simulation Speed

System Name	System Characteristics	Simulation Characteristics	Simulation Speed, Simulated days per hour
Micro Cluster	120 cores	Node sharing on	112.0
UB-HPC	8000 cores	Exclusive nodes	0.8
		Node sharing on	5.4
		Smaller bf_max_job_user	17.3
Stampede	6400 nodes		0.5

The simulator speed heavily depends on the cluster size, workload and Slurm configuration

How to Get Simulator

Various utilities and documentation are available at

https://github.com/nsimakov/slurm_sim_tools

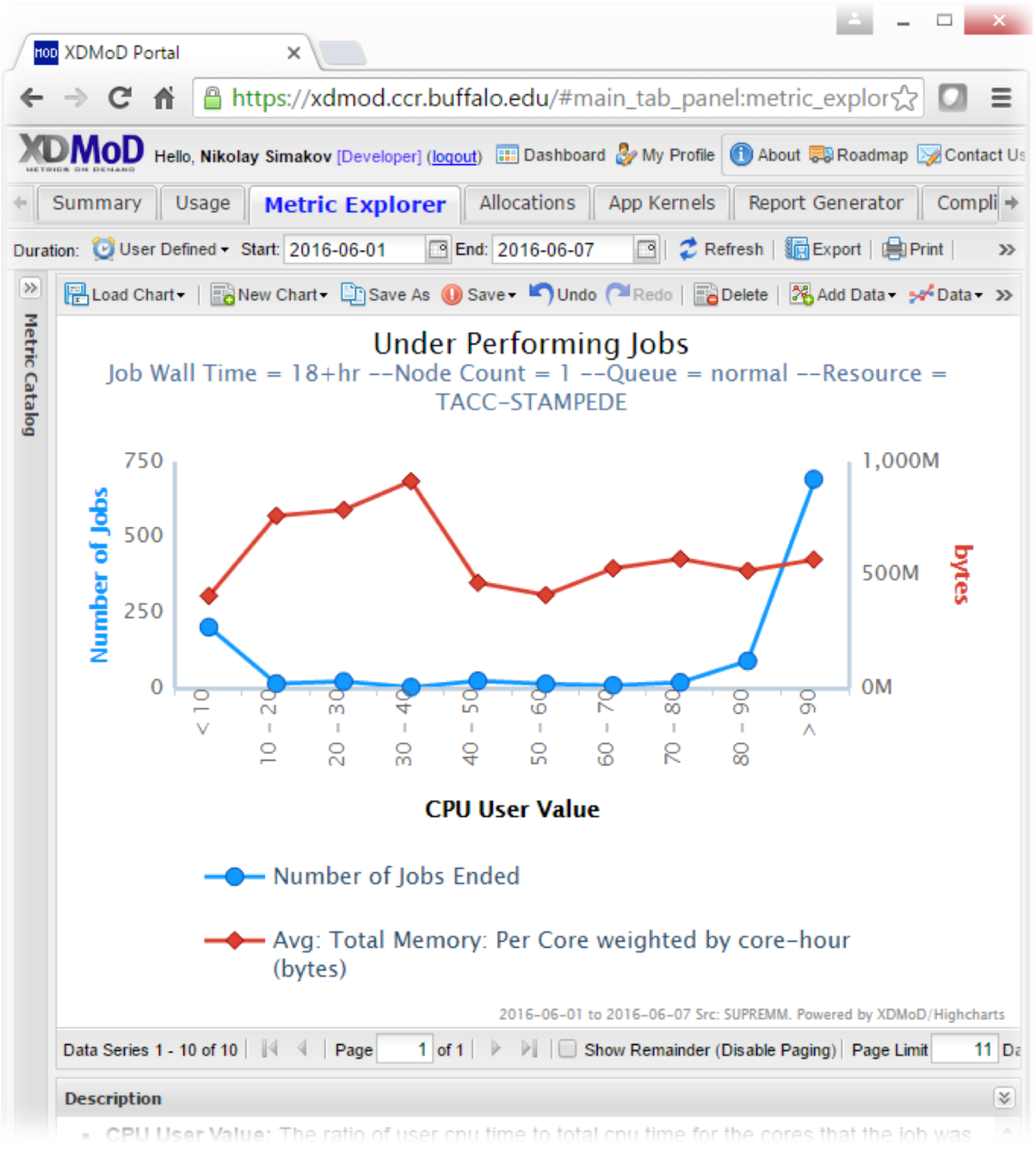
Slurm Simulator code:

https://github.com/nsimakov/slurm_simulator

Conclusions

- A new Slurm simulator was developed capable of simulation of a mid-sized cluster with a simulation speed of multiple days per hour
- Its validity was established by comparison with actual Slurm runs which showed a good match with similar mean values for job start times with a slightly larger standard deviation
- Simulator can be used to study a number of Slurm parameters that effect system utilization and throughput such as fair share policy, maximum number of user jobs considered for backfill, and node sharing policy
- As expected fair share policy alters job priorities and start times but in a non-trivial fashion
- Decreasing the maximal number of user's job considered by the backfill scheduler from 20 to 10 was found to have a minimal effect on average scheduling and decrease the backfill scheduler run time by 30%
- The simulation study of node sharing on our cluster showed a 45% increase in the time needed to complete the workload in exclusive mode compared to shared mode.
- For a large system (>6000 nodes) comprised of two distinct sub-clusters, two separate Slurm controllers and adding node sharing can cut waiting times nearly in half.

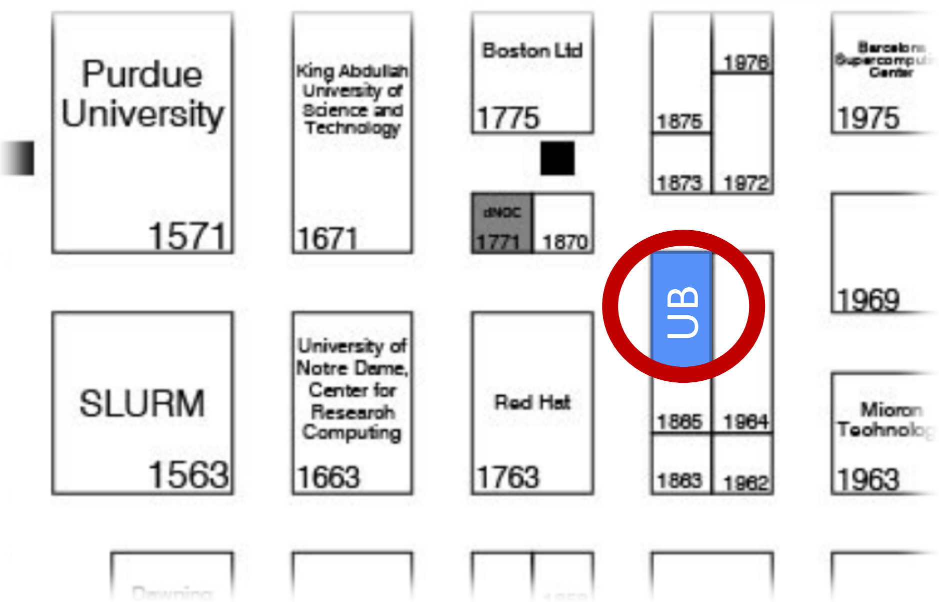
Acknowledgments



- XDMoD Developers Team
  - **UB:** Tom Furlani, Matt Jones, Steve Gallo, Bob DeLeon, Martins Innus, Jeff Palmer, Ben Plessenger, Ryan Rathsam, Nikolay Simakov, Jeanette Sperhac, Joe White, Tom Yearke, Rudra Chakraborty, Cynthia Cornelius, Abani Patra
  - **Indiana:** Gregor von Laszewski, Fugang Wang
  - **University of Texas:** Jim Browne
  - **TACC:** Bill Barth, Todd Evans, Weijia Xu
  - **NCAR:** Shiquan Su
- Funding:
  - National Science Foundation under awards OCI 1025159, 1203560, ACI 1445806.

University at Buffalo
Center for Computational Research

Questions?



- Visit our Booth at:
  - University at Buffalo/Center for Computational Research, booth 1867
- Visit XDMoD Birds-of-a-Feather (BOF) session:
  - Tracking and Analyzing Job-level Activity Using Open XDMoD, XALT and OGRT
  - Tuesday, November 14th, 5:15pm - 7pm
  - Room: 205-207

Various utilities and documentation are available at
https://github.com/nsimakov/slurm_sim_tools

Slurm Simulator code:

https://github.com/nsimakov/slurm_simulator

University at Buffalo
Center for Computational Research