Enabling and Scaling Diverse Work Loads Efficiently With Slurm

Chansup Byun, Jeremy Kepner, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, Michael Jones, Anna Klein, Peter Michaleas, Julie Mullen, Andrew Prout, Antonio Rosa, Siddharth Samsi, Charles Yee, Albert Reuther

MIT Lincoln Laboratory

SLURM User Group Meeting, September 17-18, 2019

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering.

Outline

- Introduction
- LLSC Environment
- Slurm at LLSC
  - Lua job_submit script
  - SPANK plug-in module
  - Resource limit enforcement
  - Throughput tuning
- Summary

Who We Are – a Little History


MIT Building 20

Mission: ***Development of radar systems and technology***

Main projects: Surveillance radar
Fire control radar
Navigation systems

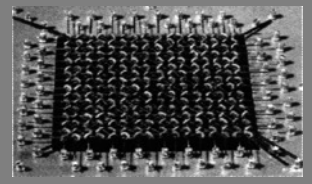
4000 employees
Designed half of all US WWII radars

Est. 1951: ***Air defense and technology development***

Main projects: Semi-Automatic Ground Environment (SAGE)

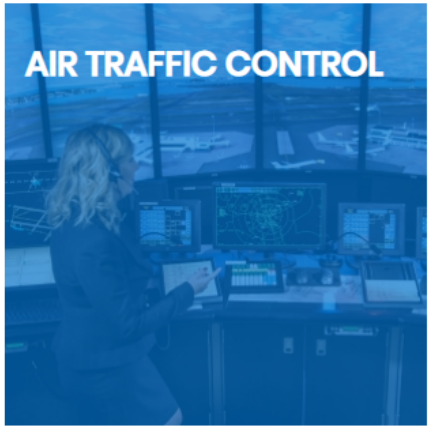
Major Innovations:

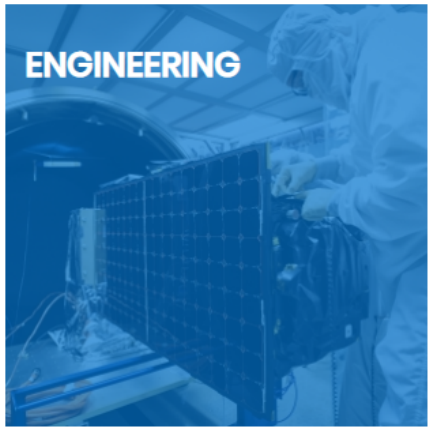
Real-Time Computing


Magnetic-core Memory


Light-pen CRT Interface

Research And Development





History of Supercomputing at Lincoln Laboratory

1951 Whirlwind

1963 Sketchpad

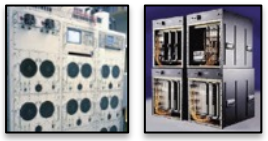
1956 TX-0

1970 Fast Digital Processor (FDP)

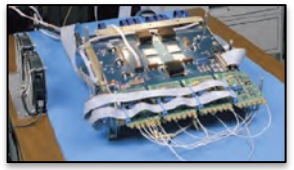
Late 1970s High-speed FFT pipelined processor

1977 : Lincoln Digital Signal Processor (LDSP)

1982 Compact LPC Vocoder



Early 1990s
- APT processor
- RAPTOR processor
- Space-Time Adaptive Processing Library (STAPL)



2002
- ISR Processing and Array Technology (IPAT) processor
- MatlabMPI

2004 Knowledge-Aided Sensor Signal Processing and expert Reasoning (KASSPER) processor

2016 Lincoln Laboratory Supercomputing Center (LLSC)

1950 | 1960 | 1970 | 1980 | 1990 | 2000 | 2010

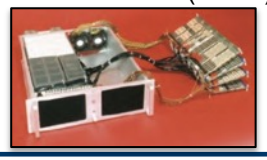
1953 Magnetic-Core Memory Array

1958
- AN/FSQ-7 (Whirlwind II)
- Average Response Computer (ARC)
- CG-24
- TX-2

1962 Lincoln Instrument Computer (LINC)

1970 GENESYS

1974 Lincoln Digital Voice Terminal (LDVT)



1978 Micro-Processor Based LPC Vocoder (LPCM)

1992 Radar Surveillance Technology Experimental Radar (RSTER) processor

1999 Parallel Vector Library (PVL)

2003 pMatlab

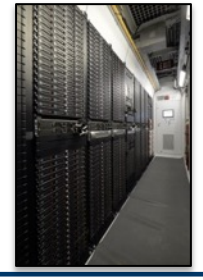
2004
- pMapper gridMatlab
- LLGrid TX-2500

2007
- Parallel Vector Tile Optimizing Library (PVTOL)
- Real-Time Communication Layer (RTCL)

2012 D4M

2015 BigDAWG

2014 LLGrid TX-Green



LLSC Approach

Approach

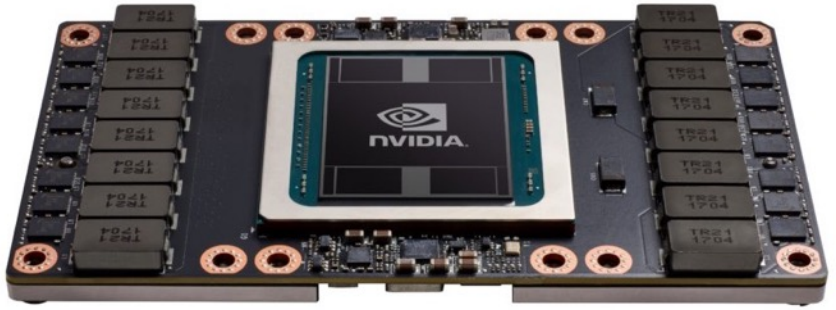
- LLSC develops and deploys unique, energy-efficient high performance computing that provides
  - Integrated HPC and Big Data capabilities
  - Data centers, hardware, software, and user support
  - 100X more productivity than standard HPC
  - 100X better performance than standard Cloud providers

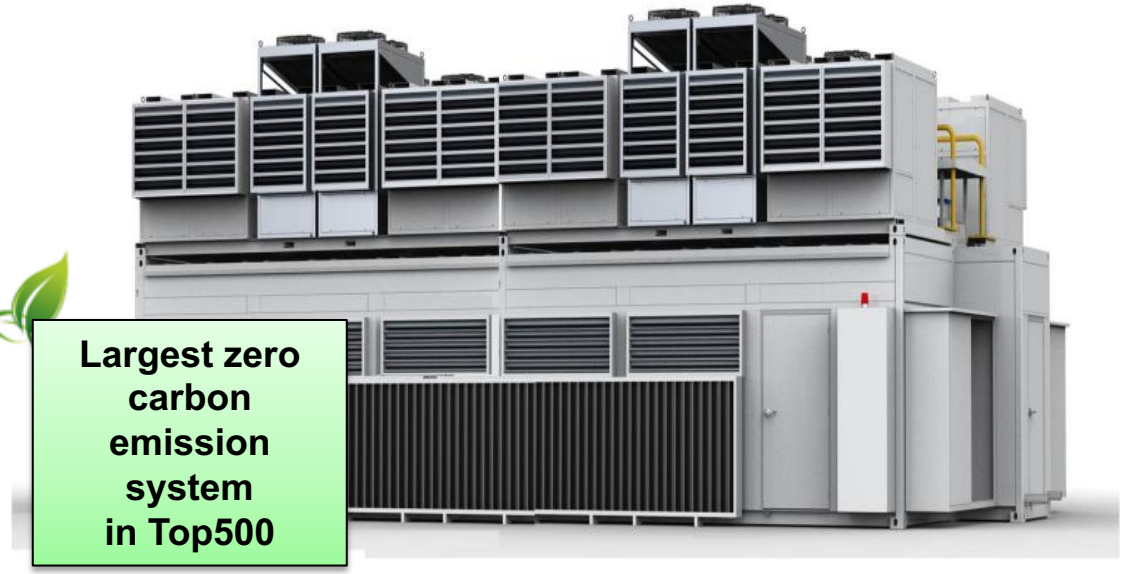
Carbon-free power

Diverse Locations

Supercomputer Upgrade & EcoPOD II

	Capability
Processor	Intel Xeon & Nvidia Volta
Total Cores	737,000
Peak	7.4 Petaflops
Top500	4.7 Petaflops (#32 in World*)
Memory	172 Terabytes
Peak AI Flops	100+ Petaflops (#6 in World*)
Network Link	Intel OmniPath 25 GB/s





Largest zero carbon emission system in Top500

- Significant increase in computing power for simulation, data analysis, and machine learning
- Leverages power of 900 Nvidia Volta GPUs
- <u>Largest AI System at any University in the World</u>

*Based on 2018 Top500.org

AI Flops = 4x4 matrix multiply half precision in, single precision out (mixed precision training)

Outline

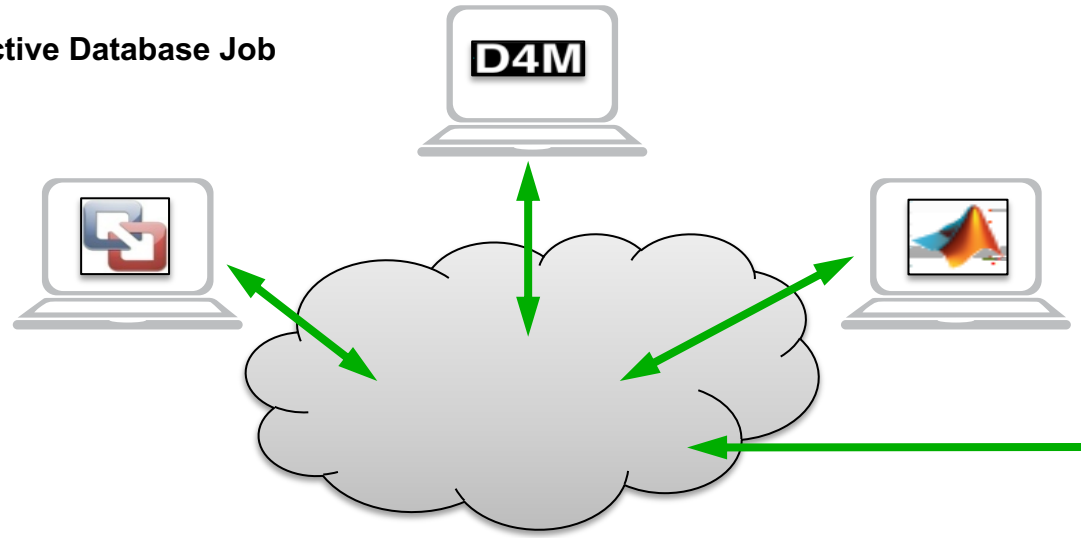
- Introduction
- LLSC Environment
- Slurm at LLSC
  - Lua job_submit script
  - SPANK plug-in module
  - Resource limit enforcement
  - Throughput tuning
- Summary

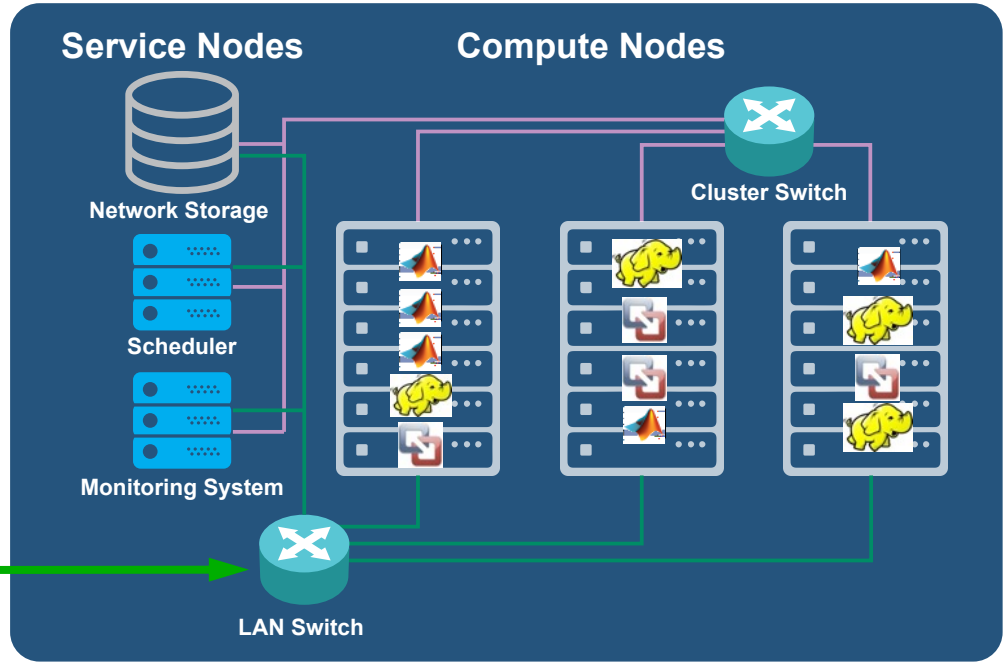
LLSC Advantage: Interactive Supercomputing

Interactive Compute Job

Interactive VM Job

Interactive Database Job

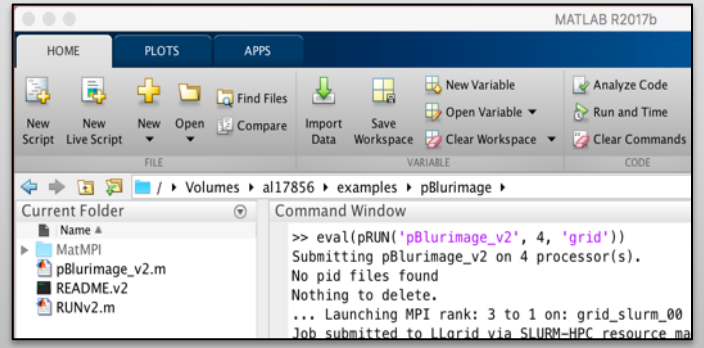




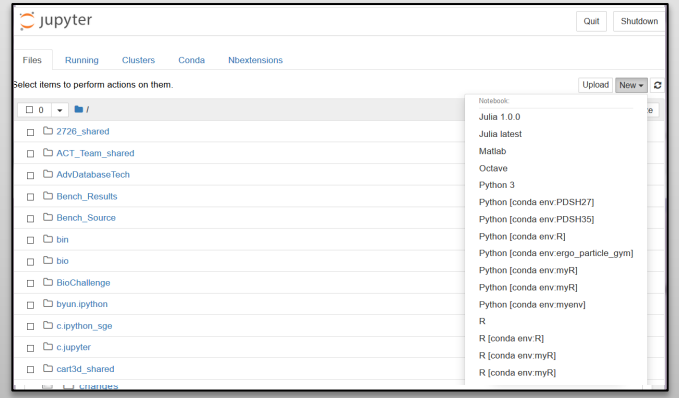
- LLSC provides a software platform that allows users to
  - Launch interactive compute jobs from their desktop
  - Share large volumes of project data
- The LLSC experience provides
  - Reference datasets pre-positioned in databases
  - Software modules and training to reduce user ramp up up time

Unique Interactive Supercomputing Capabilities

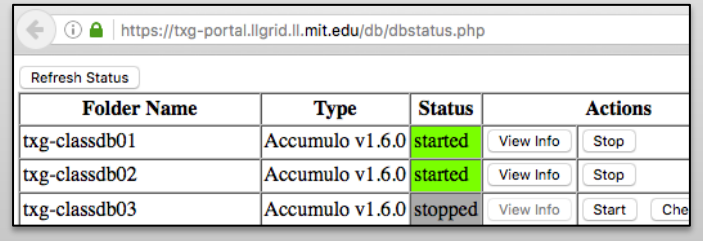
Parallel MATLAB: world's most productive parallel computing environment



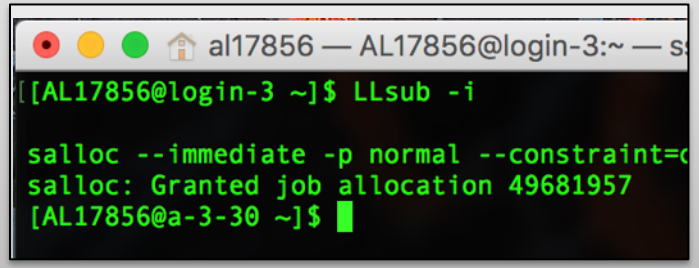
Jupyter Notebook: web-based IDE & more



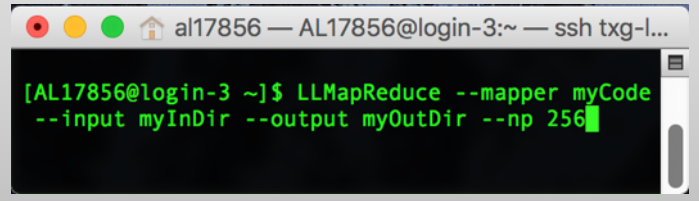
Dynamic Databases: manage world's most powerful databases from a GUI



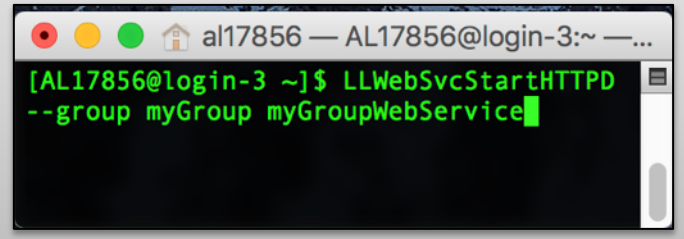
Interactive Hardware: get a processor core or a whole node



LLMapReduce: parallel data analysis in any language with one line of code



Dynamic Web Services: start an authenticated web-service



MIT SuperCloud Portal

George
Passw0rd

User-Based Firewall Rules

Connector		*Listener*
UID: 501	User is the same	UID: 500
GID: 501	Primary group is the same	GID: 702
SGIDs: • 501 • 702	Listener primary group matches one of the connector's supplemental groups	

- System services exempt
- Users configured to be exempt

gridsan/
db/
vm/
jupyter/

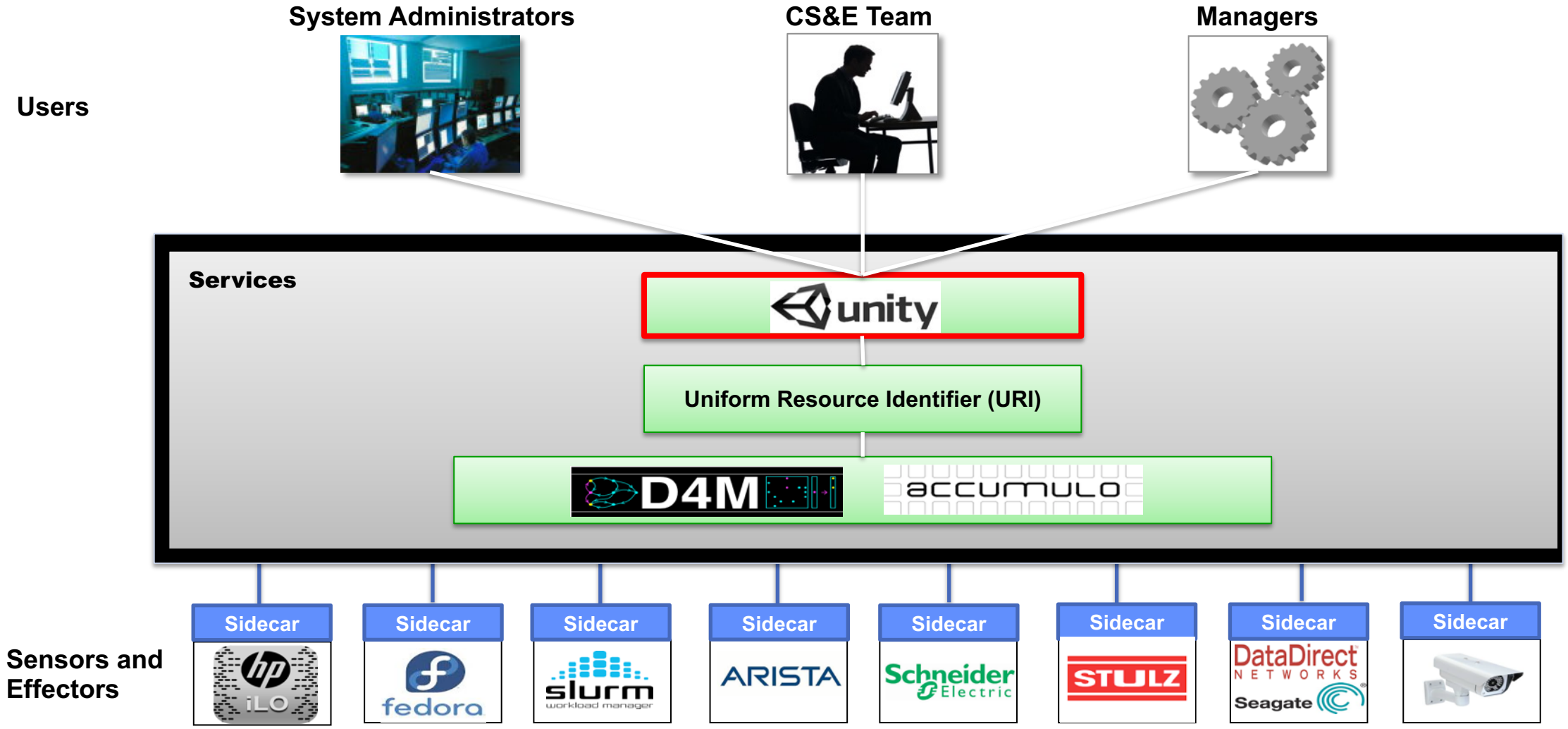
mod_dav_fs

George

Files

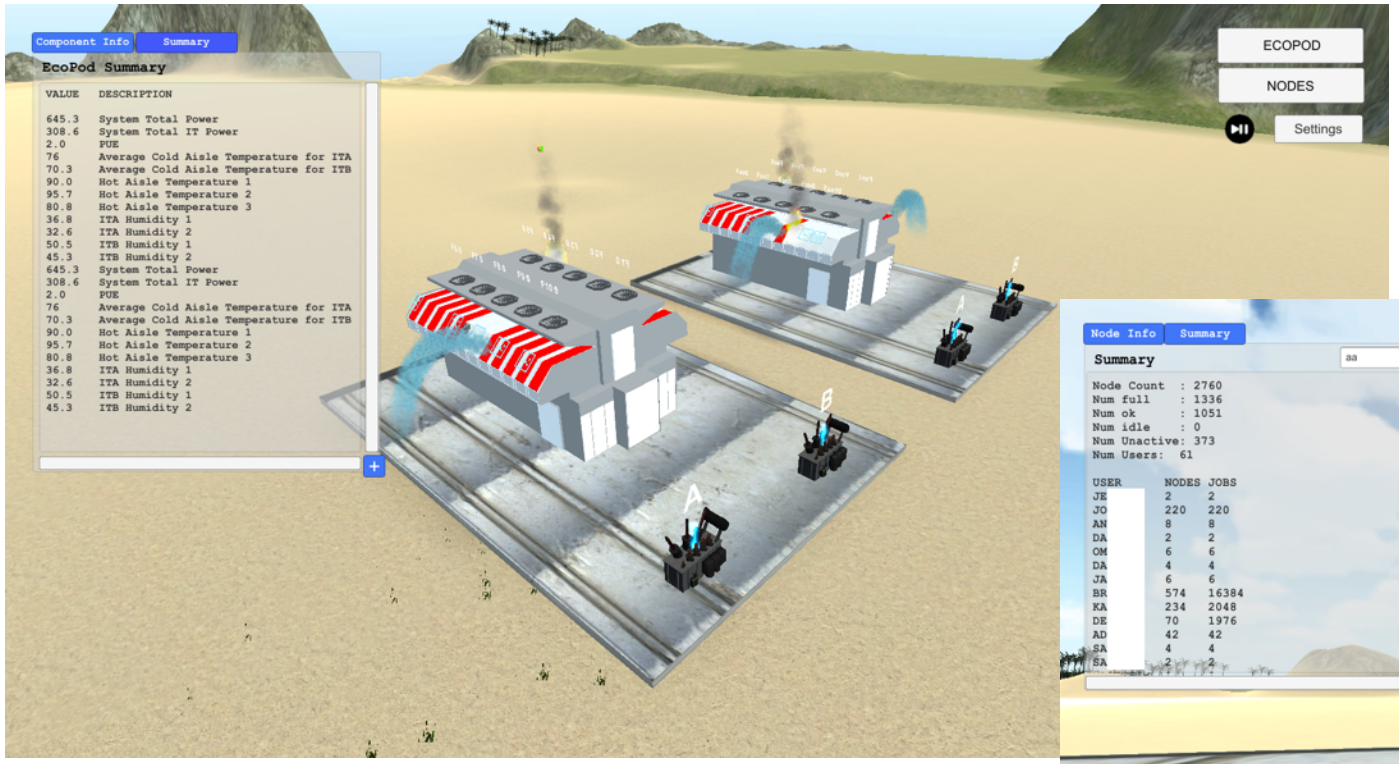
User's Desktop | Portal Node | HPC Compute Nodes and Central Storage

LLSC System Monitoring Framework



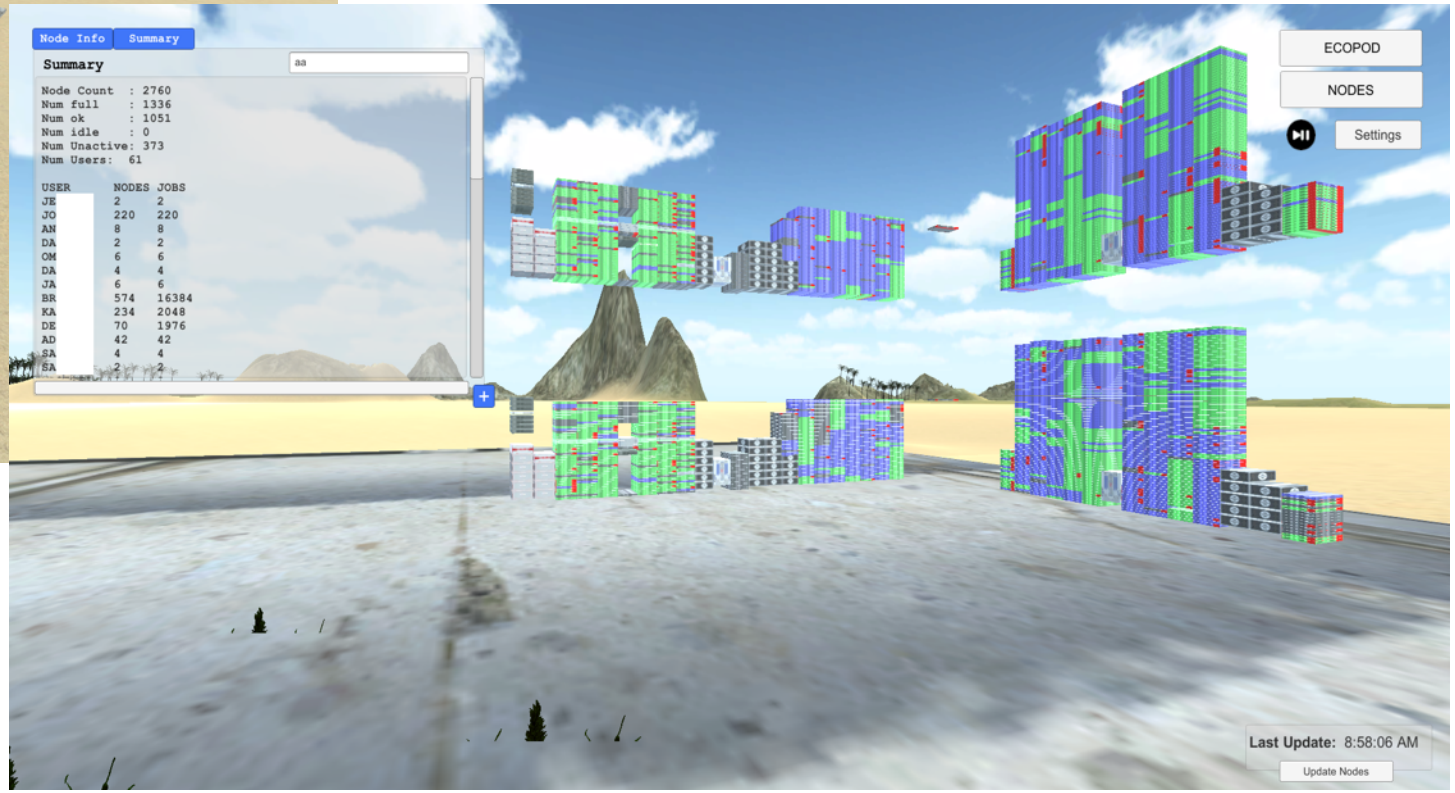
CS&E: Computational Science and Engineering

Data Center Monitoring System Screen Shots



- Data center screen shows status of all EcoPOD systems and subsystems including
  - Power distribution
  - Temperatures and humidity
  - Fans and air conditioners

- Nodes screen shows status of all login and compute nodes, storage nodes, and network switches
- Nodes are shown in rack positions



Outline

- Introduction
- LLSC Environment
- Slurm at LLSC
  - Lua job_submit script
  - SPANK plug-in module
  - Resource limit enforcement
  - Throughput tuning
- Summary

Slurm Experience at LLSC

- Slurm 15.08.8
  - Several QoS: normal, pmatlab, high, db, and gpu
  - LUA job_submit plugin
    - To enforce various requirements for jobs
  - Multi-factor priority scheduling
  - SPANK Plugin - X11 forwarding & TMPDIR
- Slurm 15.08.8 -> Slurm 16.05.10
  - Job array task dependency, “aftercorr”
- Slurm 16.05.10 -> Slurm 17.11.7
  - Native X11 forwarding support
- Slurm 17.11.7 -> Slurm 19.05.x (in progress)
  - Better support for GPU resources

Implementation Considerations

- Three separate partitions for different node types/work loads
  - Normal: General compute jobs (including DB services)
  - Manycore: Large simulations
  - GPU: Machine-Learning/AI jobs

- Support for resource management enforcement at partition level
  - User limit on resources
    - CPU: able to set different core/job slot limits within each partition
    - GPU: number based on availability and user demand
    - Memory: supports Linux OS CGROUPS kernel feature

LUA job_submit Plug-in

- Enforce the default feature (used for CPU type) request if not specified
- Set the highest QoS for interactive jobs
  - With the highest QoS and multi-factor priority scheduling, interactive jobs are immediately scheduled

```
$ salloc --immediate --constraint=opteron srun --pty bash -i
```

salloc: error: Unable to allocate resources: Immediate execution impossible, insufficient priority

```
$ salloc --immediate --constraint=opteron --qos=high \
        srun --pty bash -i
```

salloc: Granted job allocation 4109683

- Enforces GPU resource count to be 2 or 4 (Slurm recognizes one K80 as two K40s)
- GPU memory cleanup
  - GPU memory is not cleared at the end of job
  - Epilog script clears the entire K80 memory

SPANK Plug-in Module

- X11 forwarding
  - Used for applications requires a graphical user interface
  - Limited to interactive jobs only

    https://github.com/hautreux/slurm-spank-x11
  - Switched to the native X11 forwarding support with Slurm

- Redirecting TMP/TMPDIR
  - A per-job temporary directory plugin creates a directory on a local filesystem and exports it in the TMPDIR environment variable
  - This provides similar behavior of the previous scheduler
  - Useful for applications requiring a local filesystem
    - At LLSC, Lustre parallel filesystem disabled file locking for performance
  - Used for file-based message communication in gridMatlab for large scale parallel Matlab/Octave jobs[1]

[1] *Large Scale Parallelization Using File-Based Communications*, Byun et. al, IEEE HPEC 2019

LLSC Resource Limit Enforcement

- Partition specific user association limits enforced
  - normal partition: GrpTRES=cpu=512
  - manycore partition: GrpTRES=cpu=8192
  - gpu partition: GrpTRES=cpu=56,gres/gpu:tesla=16
- Allows us to adjust the per-user limit if needed
  - Users can request increased core limits for specified time periods
  - For some cases, we need to enforce the memory limit as well
- Caveat
  - User account for each partition needs to be created to enforce the partition-specific user association limit
- Desired to handle a single user account to enforce the partition-specific user association limit

Immediate Job Support

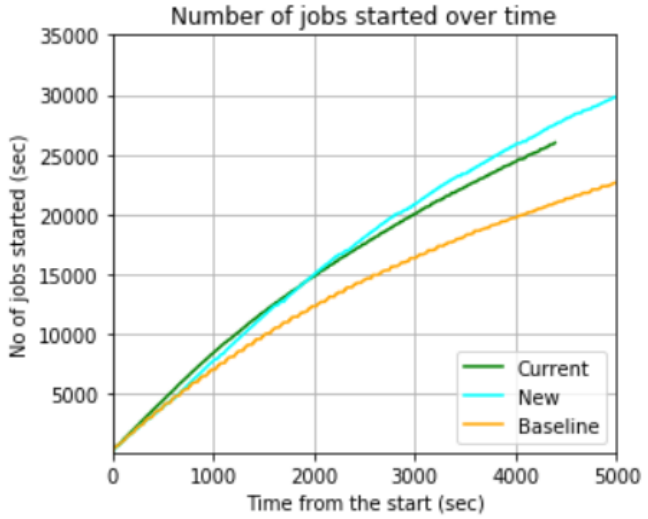
- The immediate jobs (--immediate flag) are important to LLSC users
  - Interactive, on-demand supercomputing resources for interactive jobs
  - Database jobs
  - Jupyter notebook jobs
- Multi-factor priority scheduling with a high-priority QoS
  - Configure slurm.conf for multi-factor priority scheduling
  - Attach the high QoS to all immediate jobs
- Changes with low priority jobs to harvest idle resources
  - When the cluster is full with low priority jobs, any immediate job submission is rejected
  - Add a time delay with the --immediate flag for interactive jobs
    - Time delay allows Slurm to pre-empt the low priority jobs before scheduling the immediate jobs
  - sbatch command does not support time delay with the immediate flag
    - Implemented own logic to support SPOT (low-priority preemptable) jobs

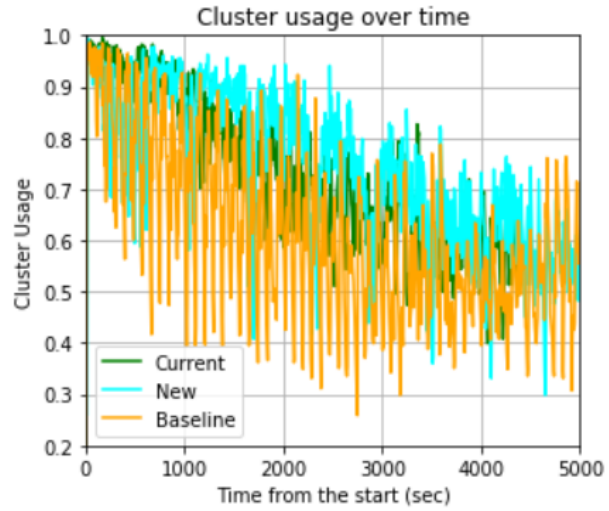
Tuning For Throughput Jobs

- **Job Characteristics**
  - **Majority of jobs are throughput jobs (array jobs)**
  - **Small number of MPI jobs**
  - **Tuning the scheduler for maximum throughput performance**
- **Tuned Parameters**
  - **Started with the baseline high-throughput configuration**

    https://slurm.schedmd.com/high_throughput.html
  - **Compared the throughput performance with the current and new parameters**

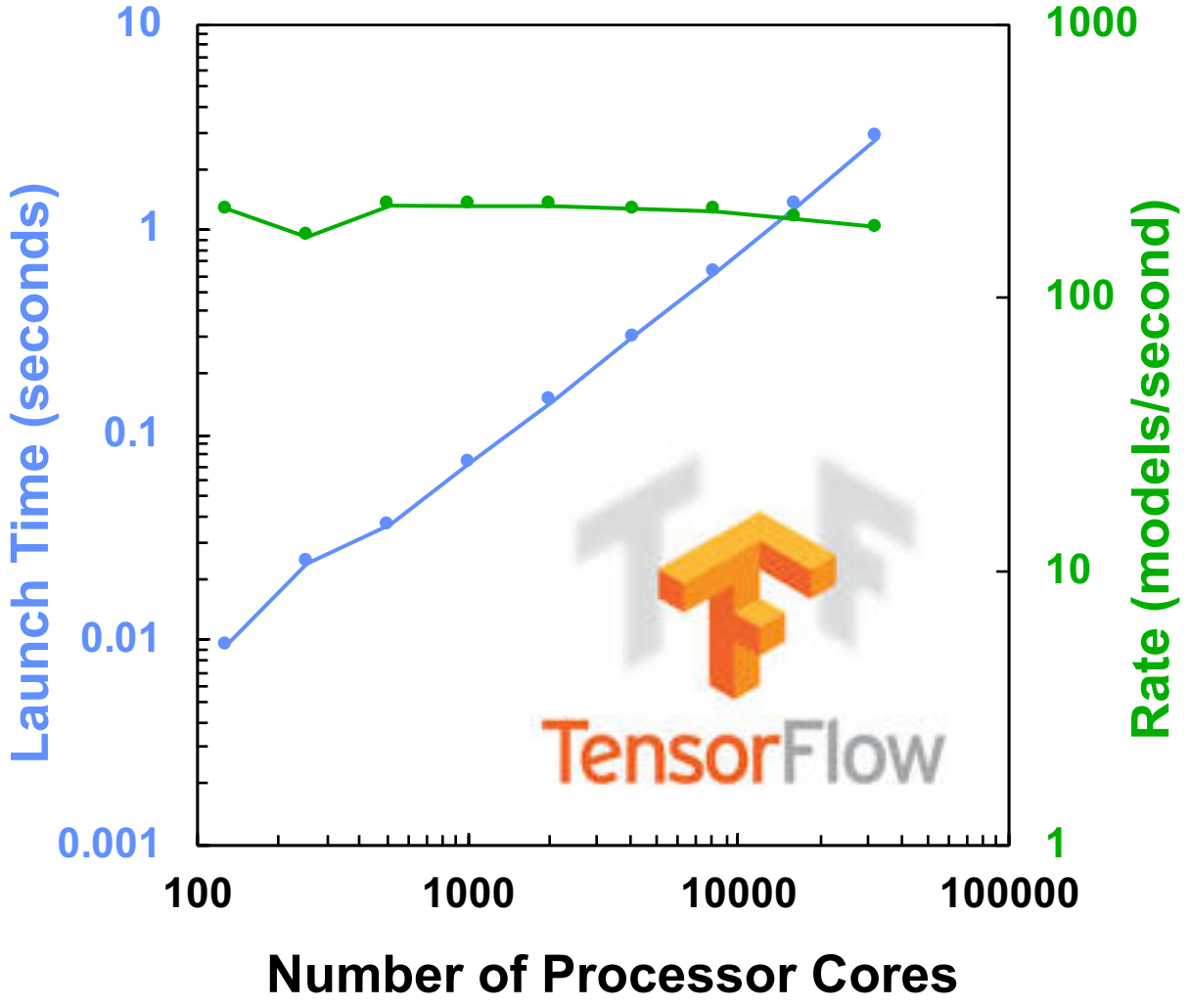
    SchedulerParameters=bf_interval=30,preempt_youngest_first,pack_serial_at_end,bf_busy_nodes,batch_sched_delay=10,bf_min_age_reserve=600,bf_resolution=600,bf_continue,bf_yield_interval=1000000,sched_min_interval=2000000,max_rpc_cnt=200





Interactive High Performance Machine Learning (HPML)
- Interactive Launch on 32,000+ Cores -

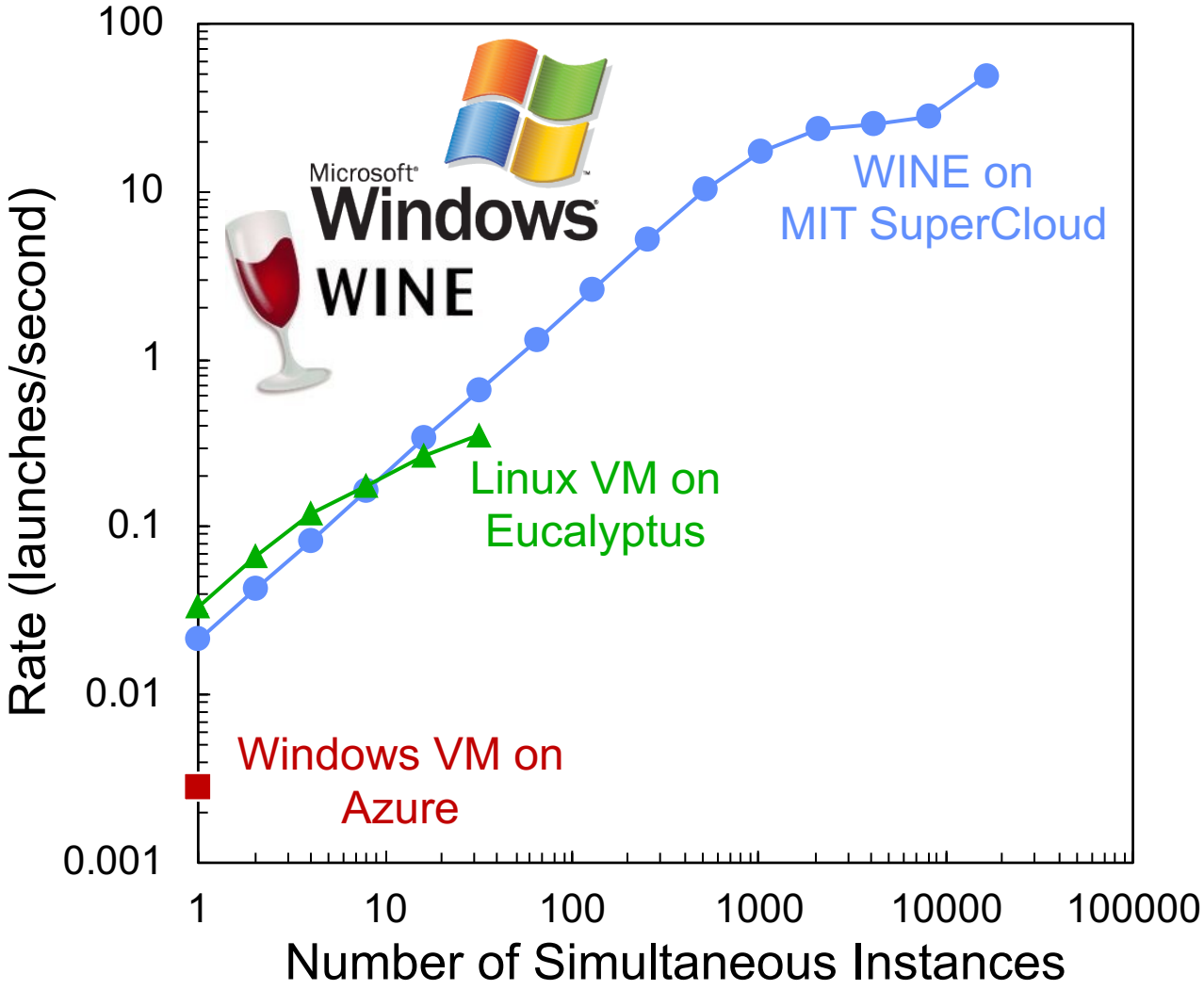
- **Machine Learning models require**
  - **High level programming environments for building models**
  - **Rapid interaction with analyst**
- **Standard approaches take minutes to hours to launch on thousands of cores**
- **MIT SuperCloud optimizes every aspect of HPML system to enable**
  - **Launching hundreds of machine learning models in seconds**
  - **32,000+ cores (512 64-core Xeon nodes)**
  - **Truly interactive machine learning**



Scalable System Scheduling for HPC and Big Data, Reuther et al, Journal of Parallel and Distributed Computing, 2017

Launching 16,000+ Microsoft Windows Environments

- **Some analytic applications are written uniquely for Microsoft Windows**
- **Standard approaches take hours to launch on thousands of cores**
  - **VMs, Windows HPC, …**
- **MIT SuperCloud optimizes every aspect of launch system to enable**
  - **16,000+ Microsoft Windows environments (running WINE)**
  - **16,000+ cores (256 x 64-core Xeon nodes)**
  - **Launched in 5 minutes**
  - **50+ launches/second**
  - **100x faster than standard approaches[1]**
  - **Truly interactive supercomputing**



[1] *Interactive Launch of 16,000 Microsoft Windows Instances on a Supercomputer*, Jones et al, IEEE HPEC 2018

Summary

- LLSC has been using Slurm for 3+ years
- LLSC has learned and exploited a number of features available to Slurm
  - LUA job_submit plug-in
  - SPANK plug-in module
  - Association limit enforcement
  - Multi-factor priority scheduling
  - QoS
  - Prolog/Epilog
  - Advance reservation
- LLSC software stack with Slurm enables scaling up users' applications
  - Interactive High Performance Machine Learning
  - Microsoft Windows Environment via Wine