

# Benchmarking and Continuous Performance Monitoring of HPC Resources Using the XDMoD Application Kernel Module

*Nikolay Simakov*

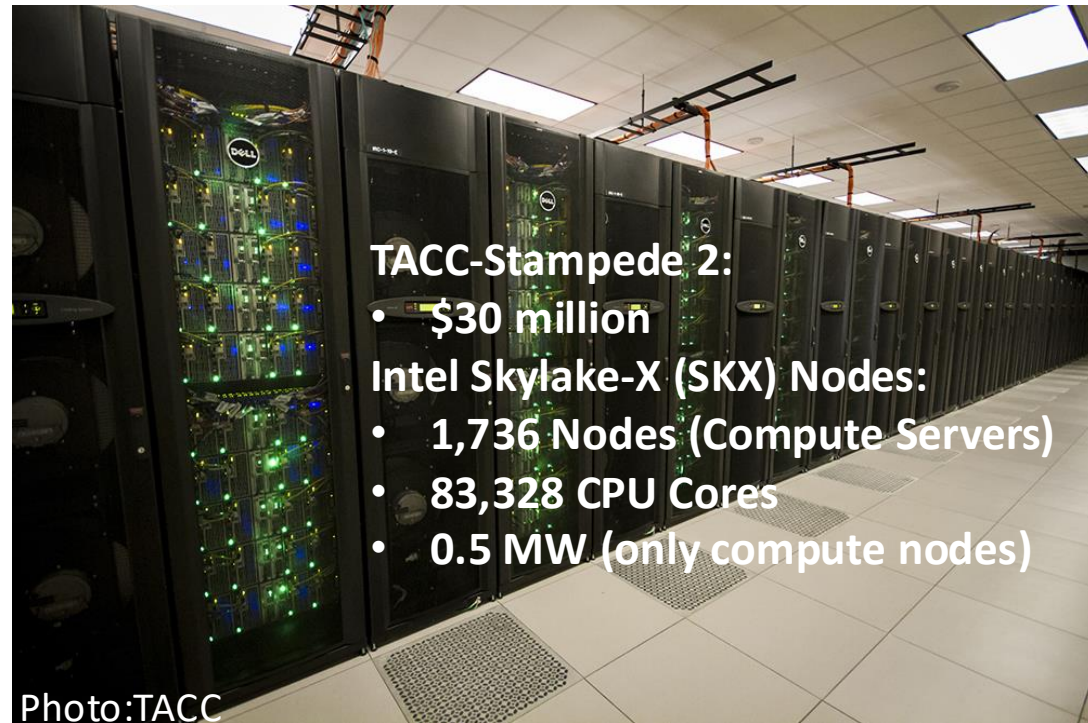
*Center for Computational Research  
SUNY University at Buffalo  
Buffalo, NY*

*November 2024*

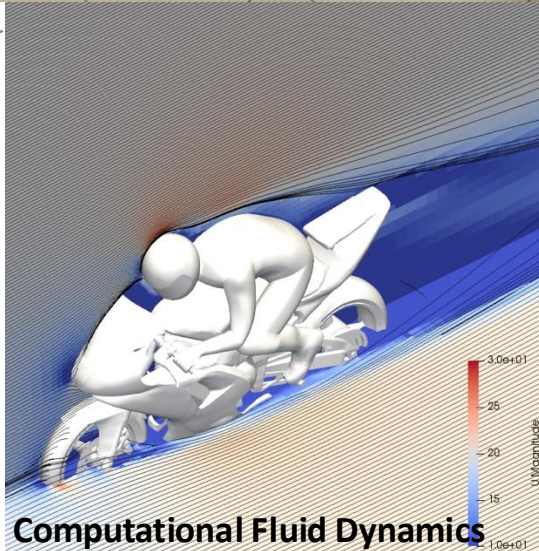
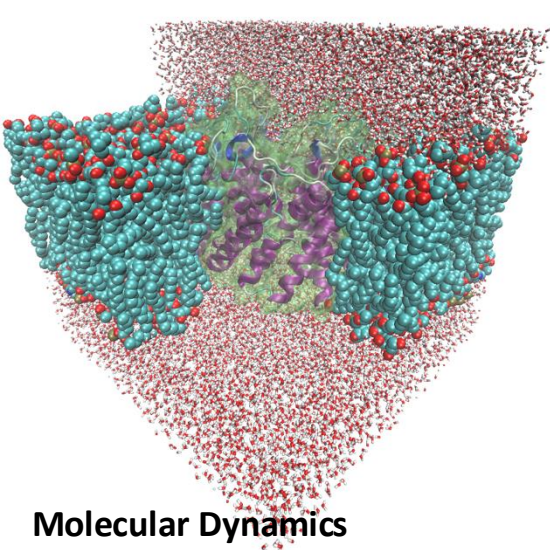
NSF Award: OCI 2137603



# Importance of Keeping HPC Resources in Optimal State



- HPC resources are often used for compute-demanding jobs in various fields of science and engineering
- HPC resources often have high utilization and long queues
- HPC resources are expensive and consume a lot of electricity



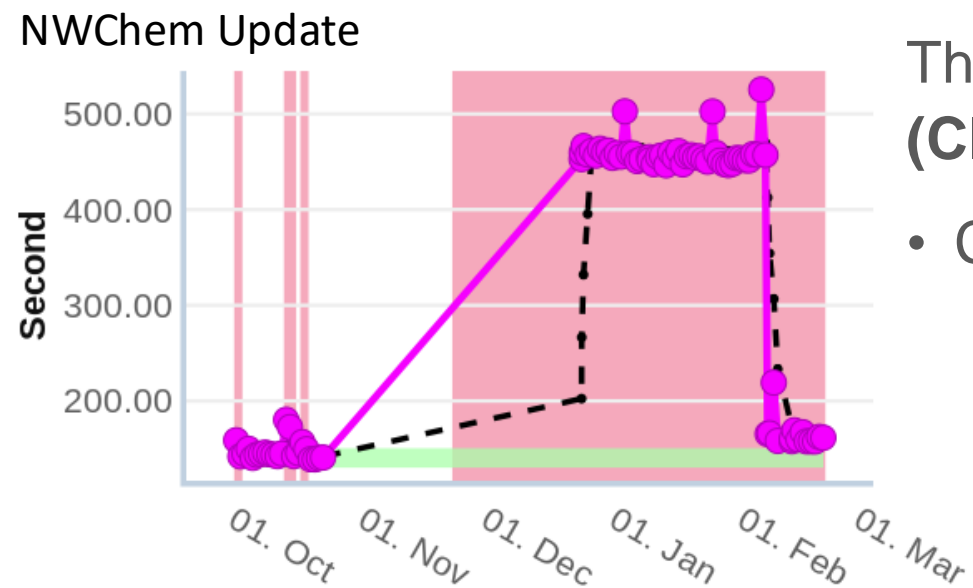
Due to the high initial and running costs and broad impact of performed science and engineering calculations, it is critical to keep HPC Resources in high-performing conditions!

# Continuous Performance Monitoring of HPC Resources



Optimal performance of applications is of high importance

- Lots of attention to performance is given at the initial deployment of HPC resource
  - Acceptance tests by vendors and sys-admins / user support
- Ideal world: benchmarking on any changes in hardware and software stack
- Reality: HPC centers are often under-staffed and benchmarking on changes are often skipped

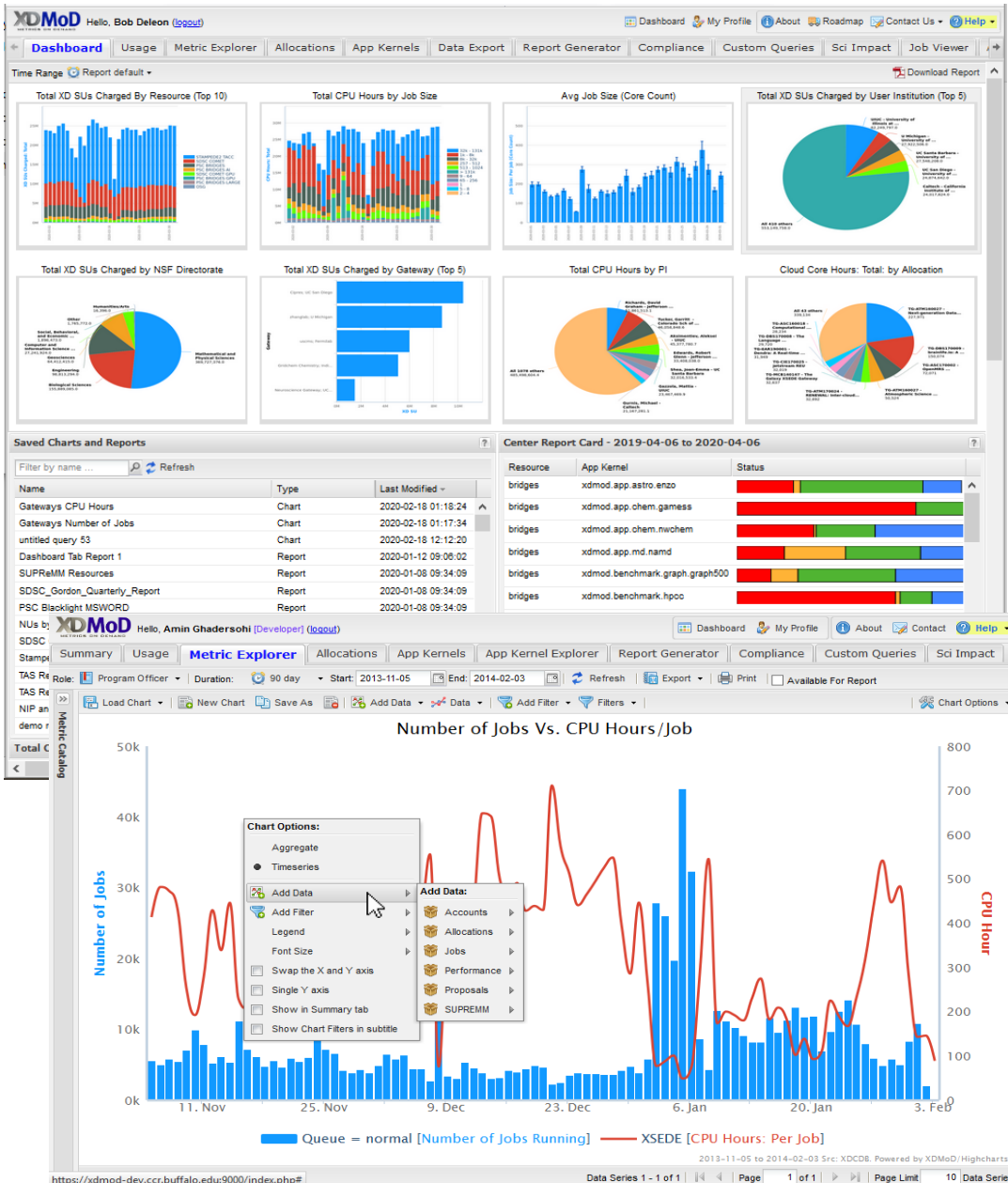


There is a need for practice similar to **Continuous Integration (CI)** in software development but for HPC resources

- Continuous Performance Monitoring:
  - Regular execution of tests based on benchmarks and real applications (ideally the same as used by users)
  - Smaller but meaningful input parameters



# XDMoD: A Comprehensive Tool for HPC System Management



## Goal: Optimize Resource Utilization and Performance

- Provide detailed information on utilization
- Continuous performance monitoring
- Enable data-driven upgrades and procurements
- Measure and improve job and system-level performance

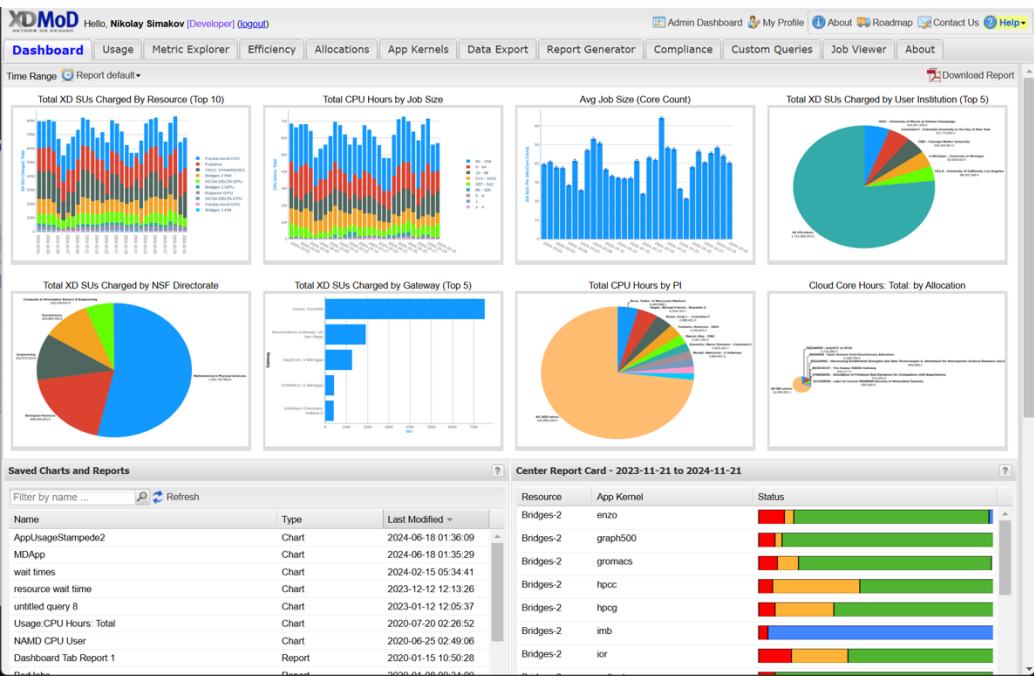
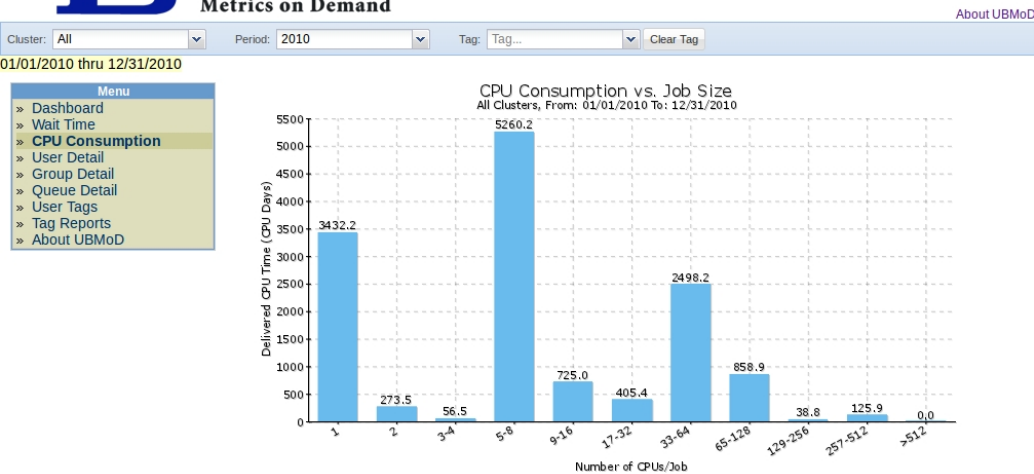
## NSF ACCESS Measurement and Metrics Service (MMS)

- Develop & deploy the **XDMoD** for monitoring ACCESS-CI

## Open XDMoD: Open Source version for Data Centers

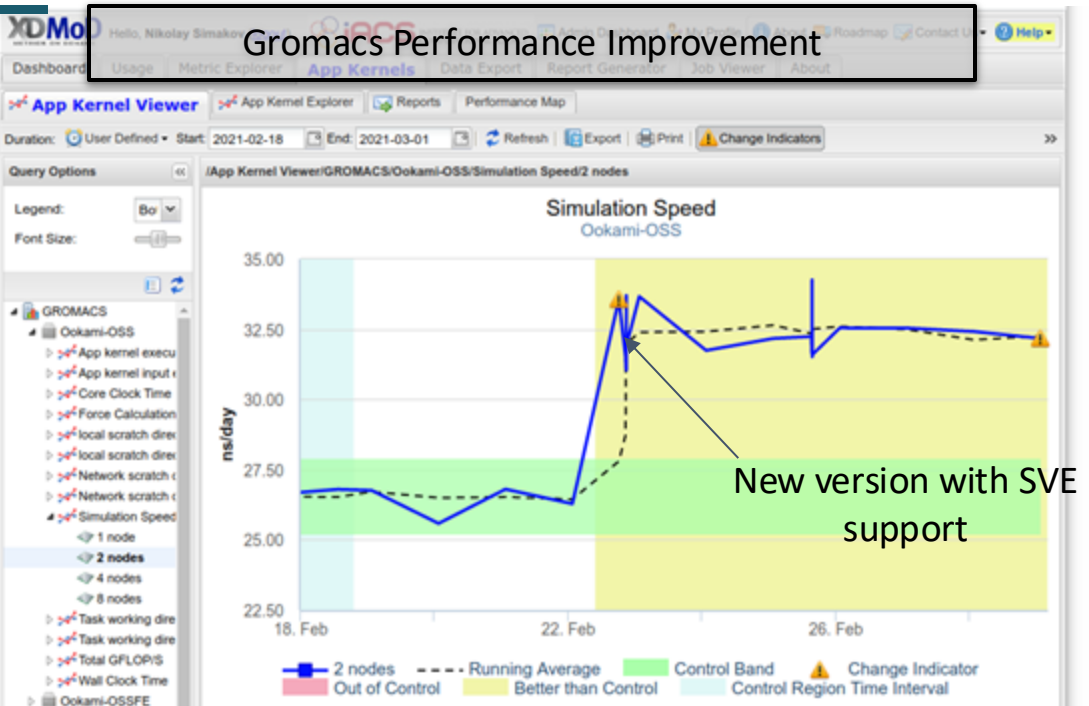
- Used to measure and optimize the performance of HPC centers
- 300+ academic & industrial installations worldwide

# XDMoD History

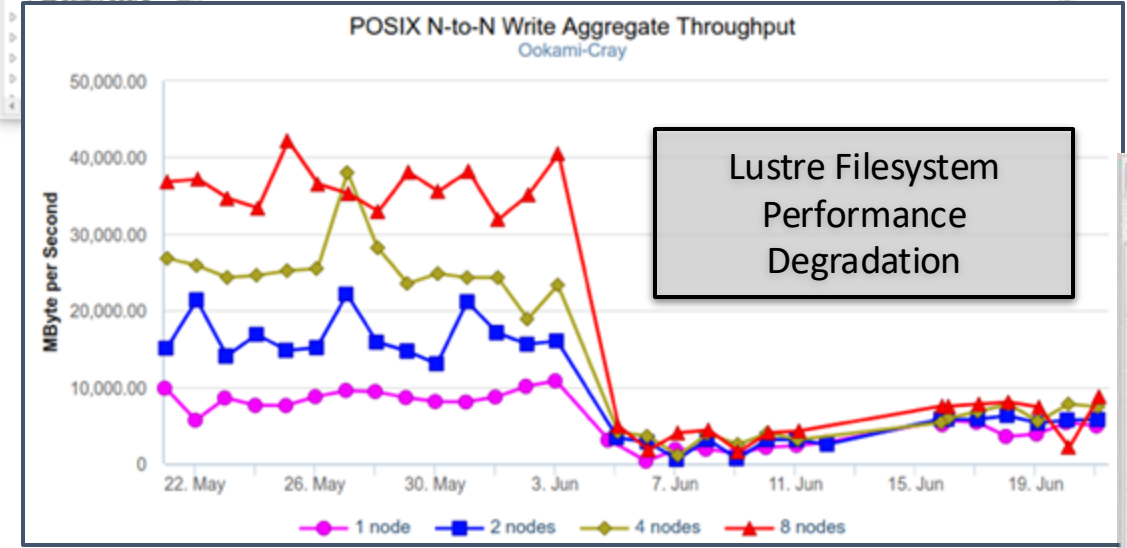


- UBMoD (UB Metrics on Demand) - tool for basic usage and accounting information visualization. 2007
- XDMoD – NSF-funded tool for monitoring XSEDE/ ACCESS cyberinfrastructure, a collection of NSF-funded HPC resources. From 2010.
  - Initially **XSEDE Metrics on Demand**
  - Then **eXtreme Data Metrics on Demand**
  - Now **XD Metrics on Demand**
- Open XDMoD – open source version of XDMoD for HPC centers. First public release 2014
- Continuous Performance Monitoring was part of XDMoD from inception
  - Referred to as “App Kernels”
  - Initially tried to leverage Inca (automated user-level testing of the software and services) for job execution automation
  - Developed Application Kernel Remote Runner for automated jobs execution on HPC resources
- **Monitoring XSEDE/ACCESS resources from 2011**
- First public release 2014

# Continuous Performance Monitoring with Application Kernels



- Application Kernels = Application + Input + Periodic execution
- Proactively identify underperforming hardware and software by regular (daily)
  - Measure Quality of Service (QoS)
- Computationally intensive but short
  - Run periodically or on demand to actively measure performance
- Measure system performance from User's view
  - Local scratch, global filesystem performance, local processor-memory bandwidth, allocatable shared memory, processing speed, network latency and bandwidth



App Kernel Viewer

App Kernel Explorer

Reports

Performance Map

Duration:

User Defined

Start: 2021-05-27

End: 2021-06-15

Refresh

Export

			May, 2021					June, 2021											
Resource	App Kernel	No...	27	28	29	30	31	01	02	03	04	05	06	07	08	09	10	11	12
Ookami-Cray	IOR	1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	U/1	F/1	U/1	U/1	U/1	U/1	U/1	U/1	U/1
Ookami-Cray	IOR	2	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	F/1	U/1	U/1	U/1	U/1	U/1	N/1	U/1	U/1
Ookami-Cray	IOR	4	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	F/1	N/1	U/1	N/1	U/1	U/1	U/1	N/1	F/1
Ookami-Cray	IOR	8	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1
Ookami-Cray	MDTest	1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1
Ookami-Cray	MDTest	2	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1	N/1

Code

Description

N Application kernel was executed within control interval

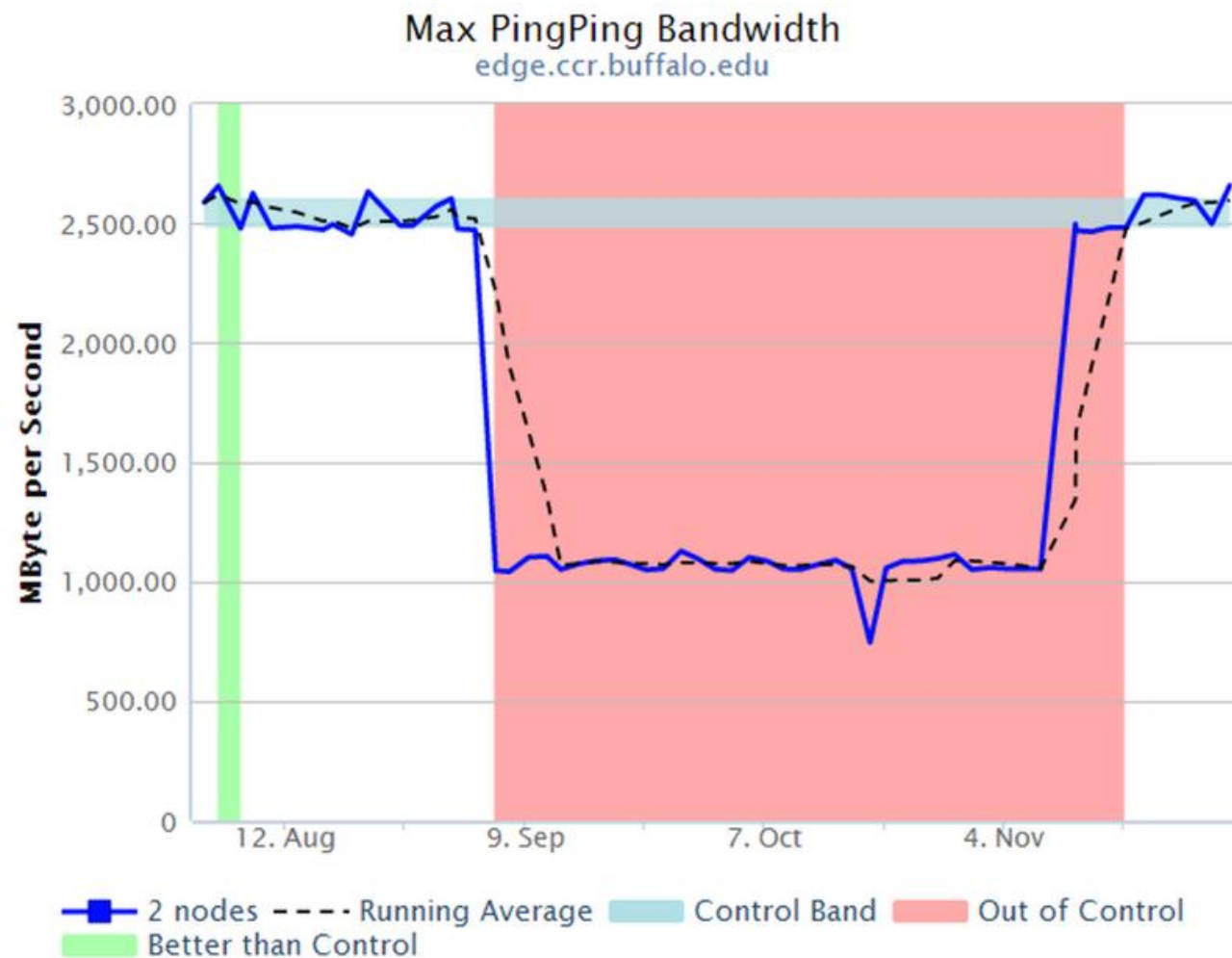
U Application kernel was under-performing

O Application kernel was over-performing

F Application kernel failed to run

# Degradation Detection: Single application

## Faulty Core Switch



The time history for the network benchmarking application kernel, IMB, demonstrates an underperforming region. The solid blue line is the data, the dashed black line is a 5-point average and the blue shading indicates the control zone range. The red zones indicate that the process is out of control in an unfavorable sense while the green zones indicate superior performance compared to the in control performance.



# Continuous Performance Monitoring with XDMoD

## XDMoD/AKRR host

### Application Kernel Remote Runner (AKRR)

- Creates, submits, and monitors batch job execution
- Parse output and push results to DB
- Daemon implemented in Python

MySQL:  
AKRR DB

### HPC Resource

- AKRR would deploy Apps inputs and some routines (app signature calculator)
- Install applications (Manually, with Spack, or use system-wide)

### XDMoD with AppKernel Module

- Ingest performance metrics
- Calculate control statistics and identify performance status
- Visualize performance
- Generate and send report

- Task – single execution of benchmark or application
  - Set to start at a specific time with desired periodicity (daily, weekly and so on)
- Supported HPC resources
  - Slurm
  - Single Linux machine
  - *PBS/Torque*
  - *OpenStack*
- Applications and Benchmarks
  - **HPCC**
  - **IMB**
  - **IOR/MDTest**
  - **Graph500**
  - HPCG
  - AI-Benchmark-Alpha
  - **NAMD**
  - Gromacs
  - **NWChem**
  - **ENZO**
  - OpenFOAM



## Master Template

```
batchJobTemplate="" "{batchJobHeader}  
{akrrCommonCommands}  
{batchJobSysDepInit}  
{akrrCommonTests}  
{akrrStartAppKer}  
{akrrCommonCleanup}  
""
```

## Resource Configuration

```
batchJobHeader="" "#!/bin/sh  
#SBATCH --nodes={akrrNNodes}  
#SBATCH --ntasks-per-node={akrrPPN}  
#SBATCH --time={akrrWallTimeLimit}  
#SBATCH --output={akrrTaskWorkingDir}/stdout  
#SBATCH --error={akrrTaskWorkingDir}/stderr  
""  
akrrPPN=128
```

## Application Configuration

```
appkernel_run_env_template = ""  
eval ` $AKRR_APPKER_DIR/execs/spack/bin/spack load  
--sh hpcc`  
EXE=$(which hpcc)  
RUN_APPKERNEL="mpirun $EXE"  
""
```

## Task Creation with CLI

```
akrr task new -r <resource> -a <appkernel> -n 1,2,4,8 -t0 "01:00" -t1 "05:00" -p 1
```

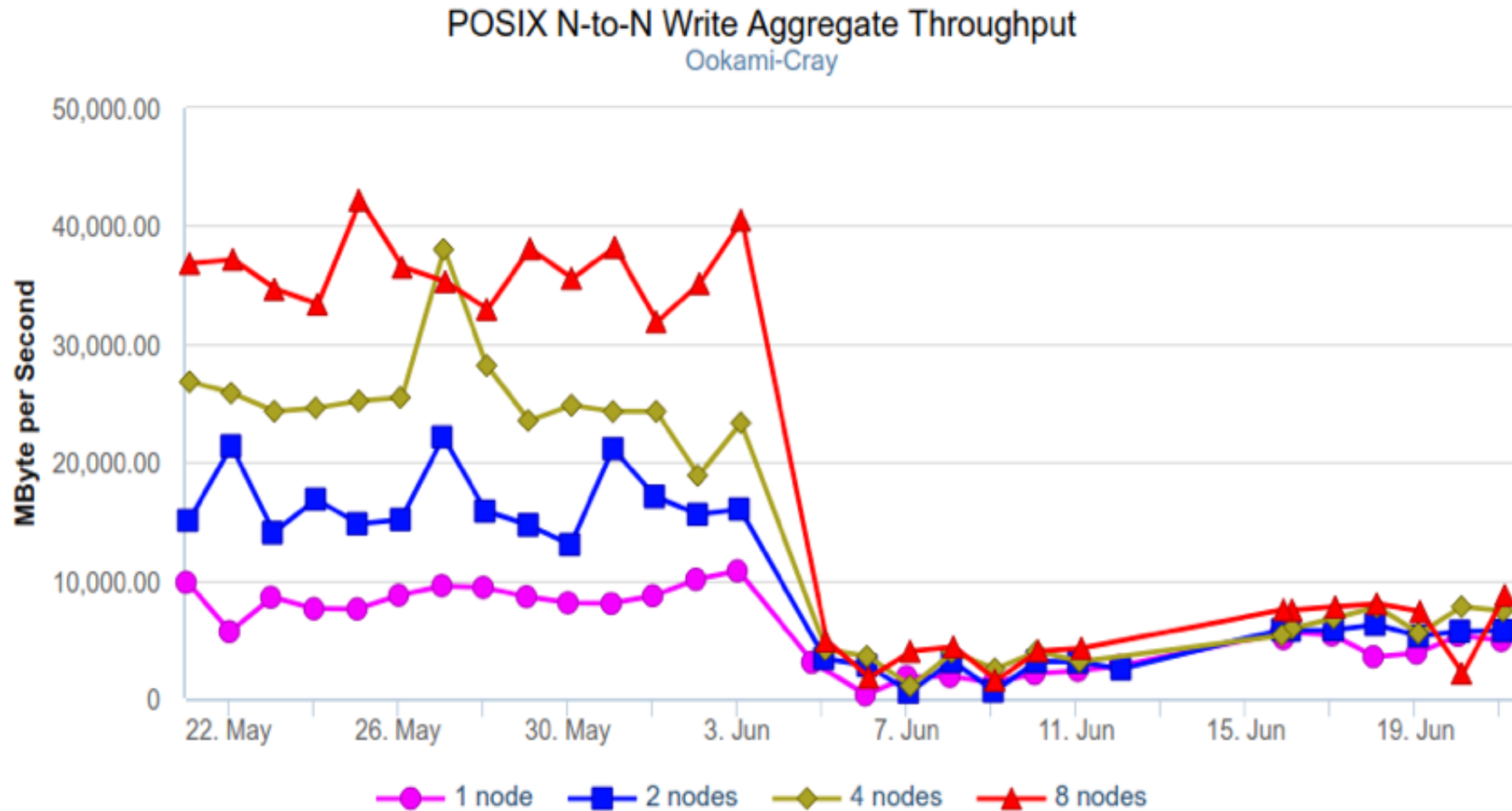
# Batch Script File Generation

- To produce a batch script, AKRR utilized a recursive string template:
  - variables in curly brackets are recursively substituted until no more change occurred
- Designed with the need to run tests on multiple resources
- Multi-level parameters hierarchy to generalize execution:
  - Default Resource Configuration
  - Default Application Kernel Configuration
  - Resource Configuration
  - Application Kernel Configuration
  - Task parameters

The background features decorative geometric patterns in the corners. The top-right and bottom-left corners are filled with a collage of shapes including triangles, circles, and semi-circles in shades of teal, yellow, and orange. Some shapes contain concentric line patterns. The bottom-right corner is a solid light blue. The text 'Use Cases' is centered in the white space.

# Use Cases

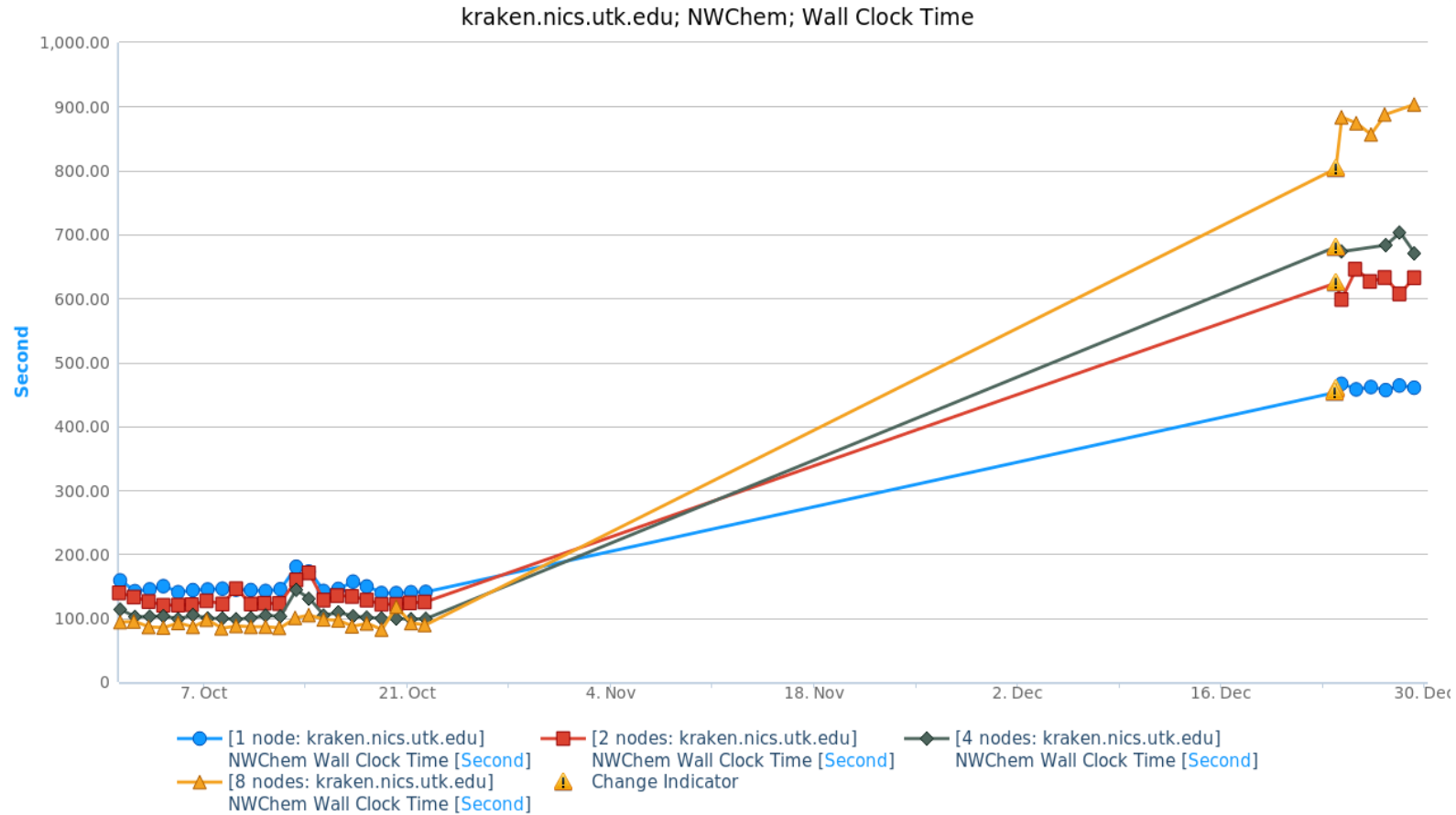
# Parallel Filesystem Performance Degradation 1



- Parallel filesystem failure leads to temporary replacement of SSD to HDD for home directory

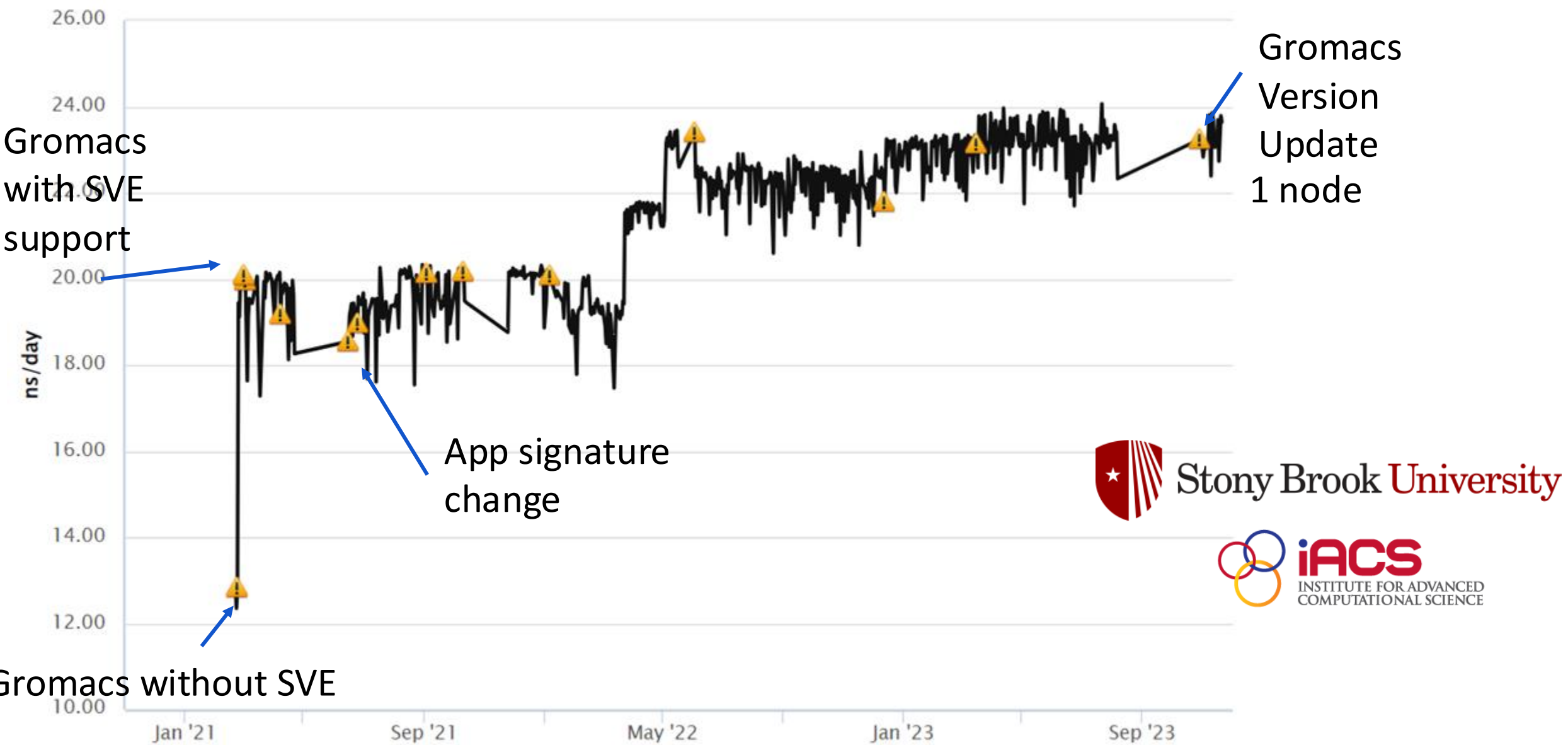


# Application Kernel Use Case

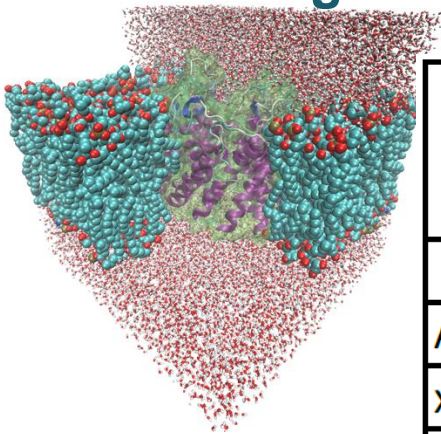


- App Kernels discover performance issues with NWChem
- Production code NWChem 6.3 is much slower than NWChem 6.1

# Monitoring Performance Improvement of Testbed: Ookami Futjisu A64FX




















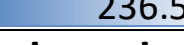

# Benchmarking New Hardware: GROMACS: Molecular Dynamics of Biomolecular Systems



CPU/GPU	Cores	Speed, ns/day (larger better)			
		MEM 82K Atoms	RIB 2M Atoms	PEP 12M Atoms	
CPU-GPU Calculations					
AMD Ryzen 9 7950X (16 Cores Used)/NVIDIA RTX 4090	16	<div><div>284.82</div></div>	<div><div>13.85</div></div>	<div><div>3.82</div></div>	
x86 (14 HT Cores Used)/NVIDIA RTX 6000 Ada	14	<div><div>245.40</div></div>	<div><div>19.30</div></div>	<div><div>2.37</div></div>	
AMD EPYC 7773X (24 HT Cores Used)/NVIDIA L40	24	<div><div>160.23</div></div>	<div><div>15.09</div></div>	<div><div>2.53</div></div>	
x86 (16 HT Cores Used)/NVIDIA H100-PCE	16	<div><div>183.92</div></div>	<div><div>15.81</div></div>	<div><div>2.88</div></div>	
Intel AVX512 Capable (16 HT Cores Used)/NVIDIA L4	16	<div><div>142.04</div></div>	<div><div>8.90</div></div>	<div><div>0.98</div></div>	
AMD EPYC 7R32 (16 HT Cores Used)/NVIDIA A10g	16	<div><div>160.53</div></div>	<div><div>8.55</div></div>	<div><div>1.76</div></div>	
Intel IceLake/NVIDIA A100	64	<div><div>242.62</div></div>	<div><div>21.41</div></div>	<div><div>2.42</div></div>	
NVIDIA Grace Hopper Superchip ES	72	<div><div>429</div></div>	<div><div>46.4</div></div>	<div><div>4.59</div></div>	
CPU Only Calculation					
ARM Fujitsu A64FX, SVE 512bit (SBU-Ookami, Fujitsu)	48	<div><div>22.8</div></div>			
ARM Amazon Graviton 3, Neoverse V1, SVE 256bit (AWS)	64	<div><div>71.4</div></div>			
x86 Intel Xeon Gold 6148, Skylake-X, AVX512 (SBU)	40	<div><div>51.40</div></div>	<div><div>4.77</div></div>	<div><div>0.42</div></div>	
x86 AMD EPYC 7643 Zen3(Milan), AVX2 (SBU)	96	<div><div>95.31</div></div>	<div><div>10.33</div></div>	<div><div>0.92</div></div>	
x86 Intel Xeon Max 9468, Sapphire Rapids, DDR mode (SBU)	96	<div><div>203.64</div></div>	<div><div>13.88</div></div>	<div><div>1.18</div></div>	
x86 Intel Xeon Max 9468, Sapphire Rapids, HBM mode (SBU)	96	<div><div>206.10</div></div>	<div><div>13.52</div></div>	<div><div>1.20</div></div>	
NVIDIA Grace CPU Superchip ES	144	<div><div>171</div></div>	<div><div>12.7</div></div>	<div><div>0.977</div></div>	



# Application Power and Performance Monitoring

CPU/GPU	Simulation Speed, ns/day	Energy Efficiency, ns/kWh
<b>ARM</b>		
Fujitsu A64FX, SVE 512bit	 22.8	 9.1
Amazon Graviton3, Neov.V1, SVE256	 71.4	
Ampere Altra, Neoverse N1	 56.5	 8.7
<b>x86 AMD</b>		
EPYC 7742 Zen2(Rome), AVX2	 109.6	
EPYC 7763 Zen3(Milan), AVX2	 169.9	
<b>x86 Intel Xeon</b>		
Plat. 8160, Skylake-X, AVX512	 70.4	
Plat. 8380, Ice Lake, AVX512	 133.3	
Gold 6130, Skylake-X, AVX512	 39.3	 4.5
Gold 6330, Ice Lake, AVX512	 81.7	 5.4
Gold 6330, Ice Lake, AVX512 (ICC)	 103.0	 6.9
<b>x86 Intel Xeon with NVIDIA GPU</b>		
Gold 6130, NVIDIA V100	 145.1	 13.9
Gold 6330, NVIDIA A100	 236.5	 13.9

- **GROMACS performance on selected platforms**

- Measure both performance and power consumed on a per application basis
  - Provide analysis of time to solution versus power consumed for applications
- **Useful for new and emerging technologies**
  - Will the energy savings of energy efficient architecture be offset by longer running times for the job mix?
- Benchmarking extended to systems that are available on public clouds using ACCESS CloudBank
- **Application developers can tune applications to achieve architecture specific energy efficiency**

# Future Plan

- Can we do continuous performance monitoring without affecting users' jobs?
  - Hypothesis: small benchmarking jobs can fall into gaps between users' jobs
  - Based on the historical workloads, generated workload for study effects of different resource utilization and test the effects with Slurm Simulator and HPCMod
- Automate energy metrics addition
- Move analytics to AKRR from XDMoD-AppKernel
- Add other benchmarking tools as data-source to XDMoD

# Conclusions

- The Continuous Performance Monitoring Module of XDMod automates the identification of performance degradation by regularly executing a set of applications and benchmarks.
- The input data are carefully designed to yield crucial performance insights within a short wall time, enabling efficient monitoring.
- The module incorporates an algorithm that detects performance degradation
- The email reports can be sent regularly or upon detecting performance issues.



# XDMoD Teem

- University at Buffalo - Center for Computational Research

- Tom Furlani



- Matt Jones

- Bob DeLeon



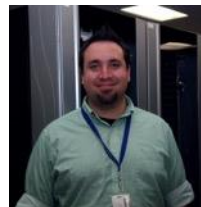
- Joe White



- Nikolay Simakov



- Ryan Rathsam



- Conner Saeli



- Andrew Stoltman

- Aaron Weeden



- Gregory Dean

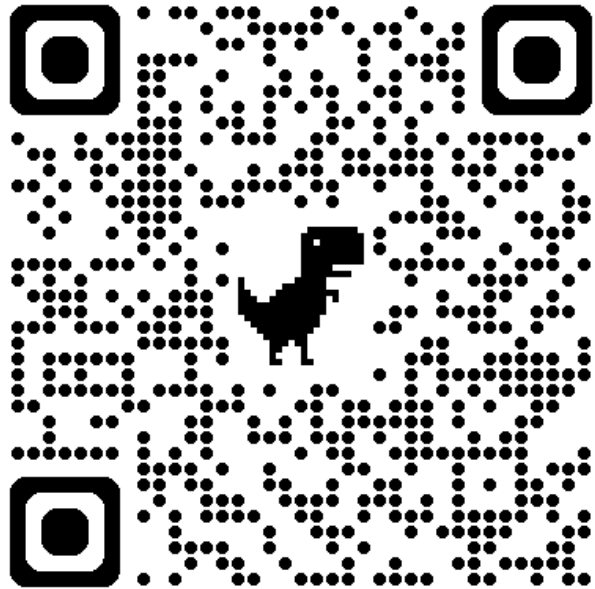


- Josh Furlani

# Acknowledgements



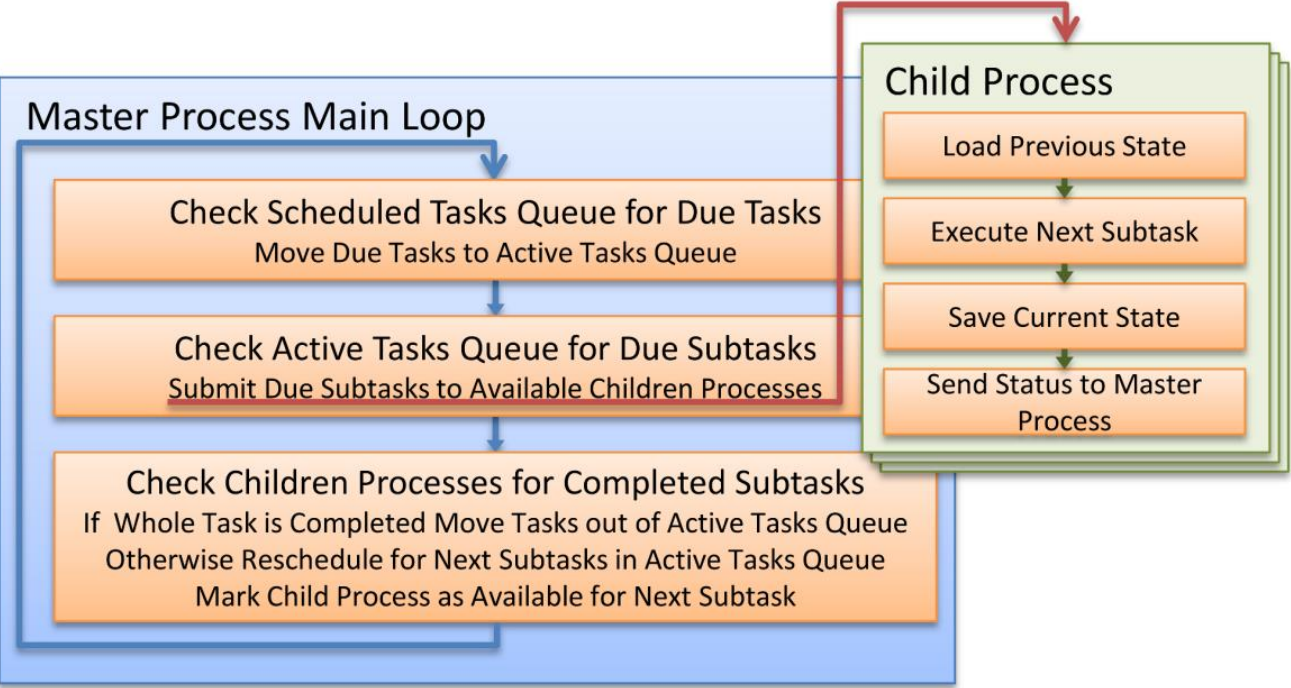
NSF OAC Awards: 2137603



Checkout our Slurm Simulator at <https://github.com/ubccr-slurm-simulator>

- This work is supported by the National Science Foundation under award 2137603.
- This work used compute resources at Stony Brook University iACS, SUNY University at Buffalo CCR, the XSEDE/ACCESS and CloudBank
- Presentation at <http://dx.doi.org/10.13140/RG.2.2.13362.62409>
- See <https://github.com/ubccr/akrr> and <https://open.xdmod.org/> for more details.

# AKRR Daemon Design



Scheduled Tasks Queue

Due Time	Resource	App.Kernel	N <sub>nodes</sub>	Repeat in
2014-08-10 01:15	Stampede	NAMD	8	1 day
2014-08-10 01:15	Trestles	NWCHEM	4	1 day
2014-08-11 05:45	Gordon	GAMESS	4	3 day
2014-08-11 01:15	Lonestar	HPCC	16	1 day

Active Tasks Queue

Due Time	Resource	App.Kernel	N <sub>nodes</sub>	Status	Next Step
2014-08-10 01:10	Stampede	NAMD	4	New Task	Create and Submit to Queue
2014-08-10 01:12	Gordon	NWCHEM	8	Still in Queue	Check the Queue

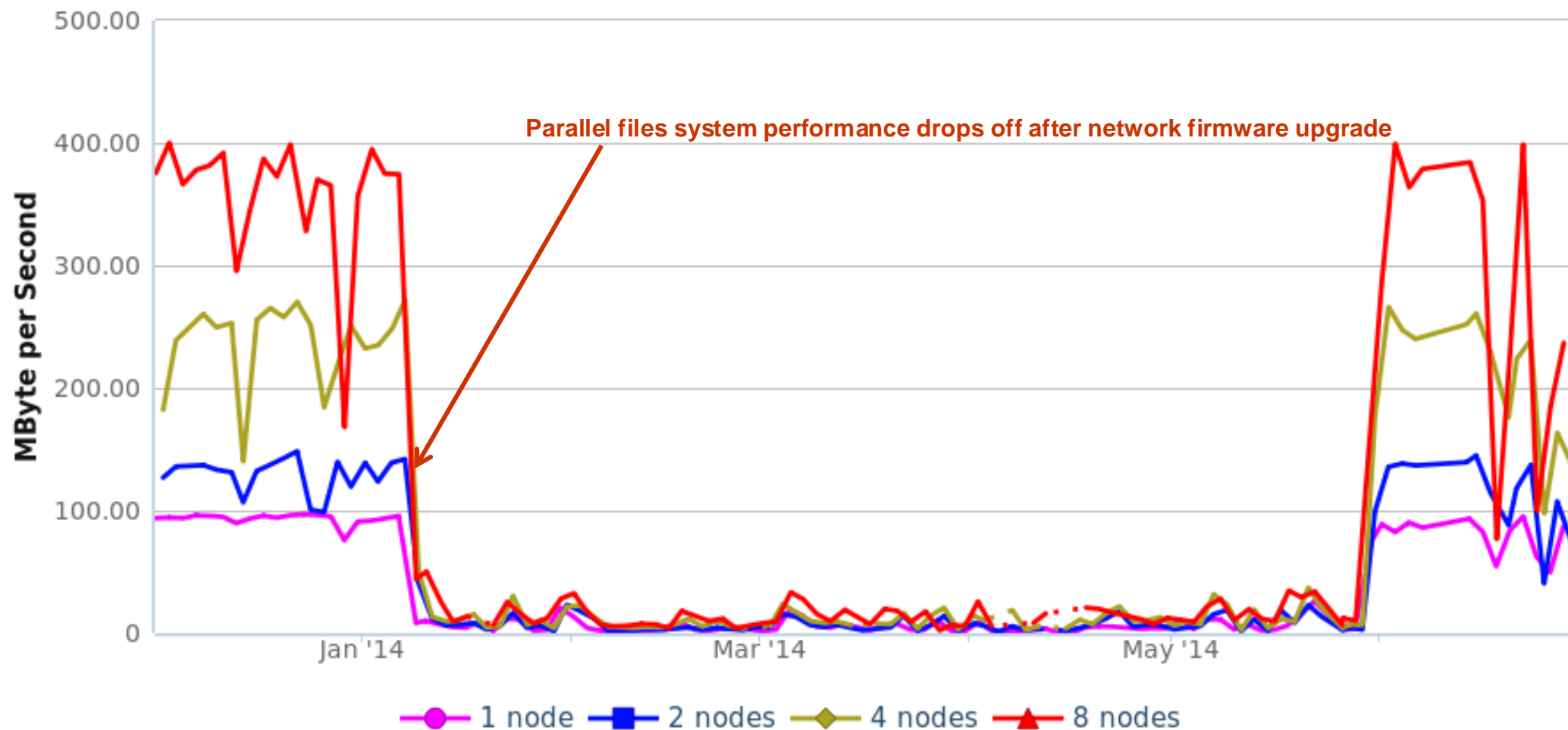
AKRR (Application Kernel Remote Runner) master process main loop with example scheduled and active task queue content.



# Parallel File System Degradation

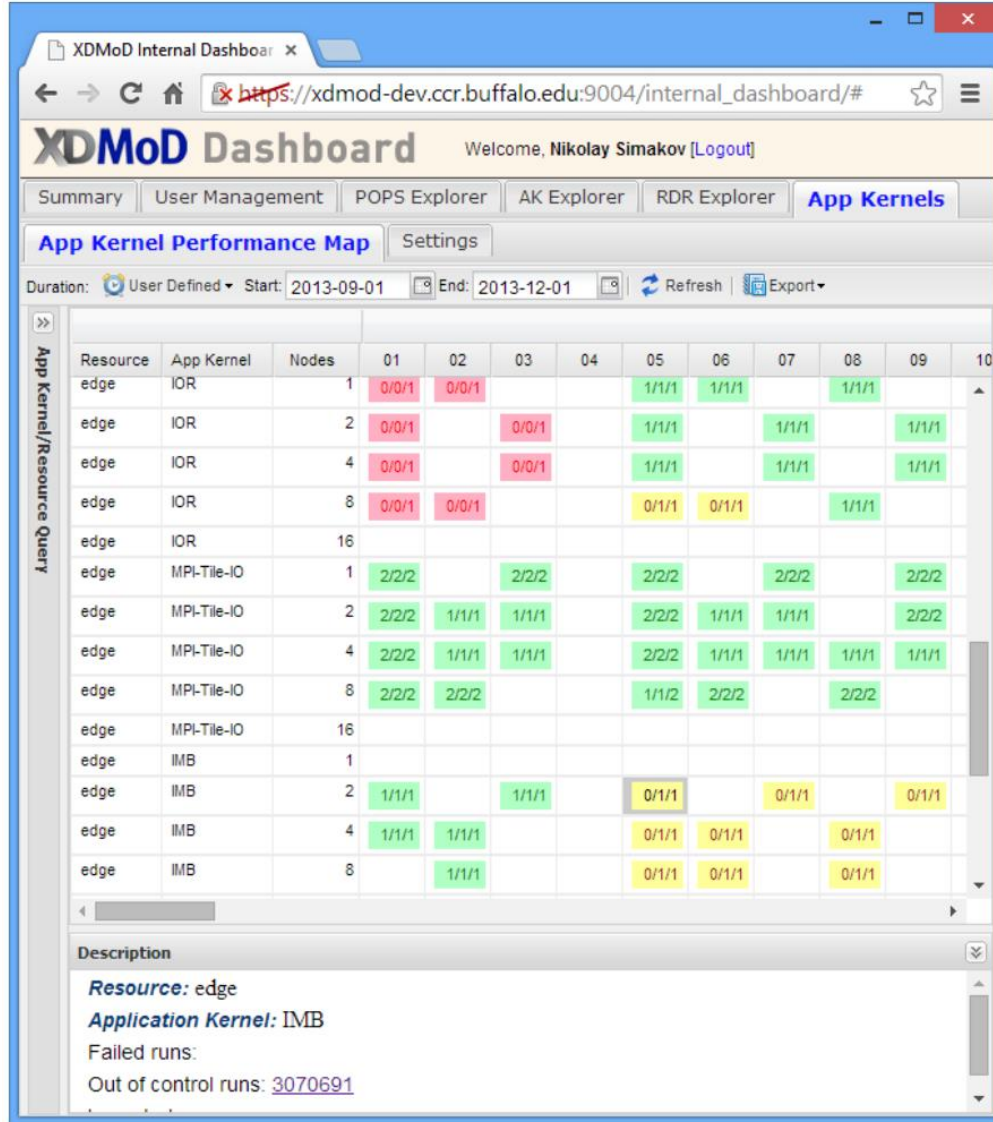
MPIIO Collective N-to-1 Write Aggregate Throughput

[edge.ccr.buffalo.edu](http://edge.ccr.buffalo.edu)



2013-12-01 to 2014-06-30 Src: XDMoD App Kernels. Powered by XDMoD/Highcharts

# Degradation Detection: Performance map



A portion of the application kernel performance map for the Edge cluster at CCR-UB showing which kernels ran normally (green), which ones ran sub-optimally (yellow), and which ones failed to run (red).