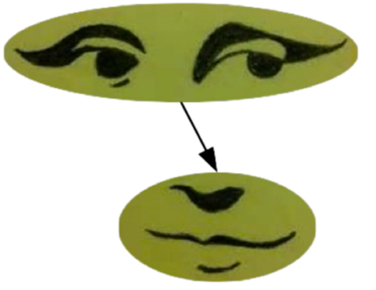
Performance Monitoring and In Situ Analytics for Scientific Workflows

Allen D. Malony, Xuechen Zhang, Chad Wood, Kevin Huck

University of Oregon

9th Scalable Tools Workshop

August 3-6, 2015



Talk Outline

- A whole bunch of motivation
- Scientific workflows (more inspiration than motivation)
  - What are they?
  - Productivity, scientific productivity, exascale productivity
  - Future scientific workflows
- MONA project
- WOWMON (WOrkfloW MONitor)
  - Design and prototype
  - Demonstration
    - LAMMPS
    - GTS
- Next steps

Scientific Workflows

- Workflows for scientific investigation
- Capture scientific methodologies and processes
  - Experimental measurement (multiple experiments)
  - Computational simulation (multiple simulations)
  - Measurement and simulation data analytics and visualization
  - Capture of provenance (metadata)
  - Multi-experiment data repositories
- Automation of scientific methodologies and processes
  - Workflow creation and execution
  - Usability and reproducibility
- Apply computer science methods, tools, and technologies to increase *scientific productivity*

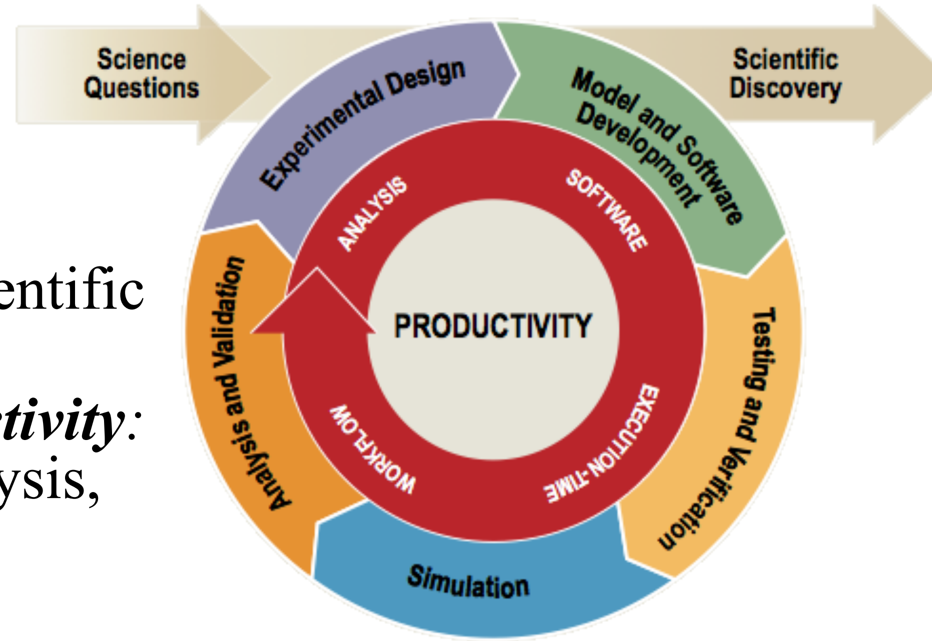
*Productivity – a Computing Metric of Merit**

- Rich measure of quality of the computing experience
  - Captures key factors that determine overall impact
  - Greater productivity, better computing experience
- Productivity is strongly related to ease of use
  - Less effort for same result in same time
- Expands our notion of computing effectiveness
  - Focuses attention on important effectiveness contributors
  - Exposes relationships between
    - program development and program execution
    - time to develop/maintain/configure/… with time to solution
- ***Productivity unifies usability and performance***
  - Expresses tradeoff between
    - programmability and delivered performance

* Courtesy of Thomas Sterling, Indiana University

HPC is about Scientific Productivity

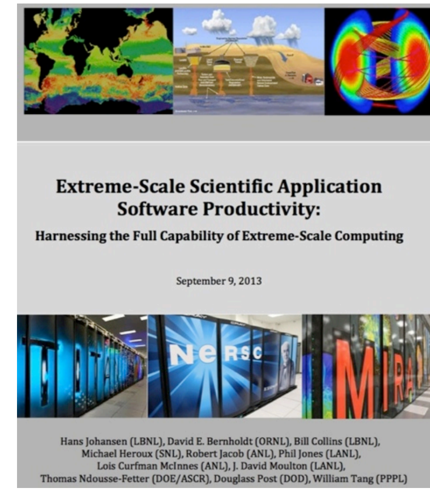
- *Scientific productivity* is a quality measure of the process of achieving science results, incorporating:
  - ***Software productivity***: development effort, time, maintenance, support
  - ***Execution-time productivity***: efficiency, time, cost to run scientific workloads
  - ***Workflow and analysis productivity***: experiment design, results analysis, validation, hypothesis testing
  - ***End-to-end productivity:*** from science questions to scientific discovery (i.e., *value* of scientific insights)



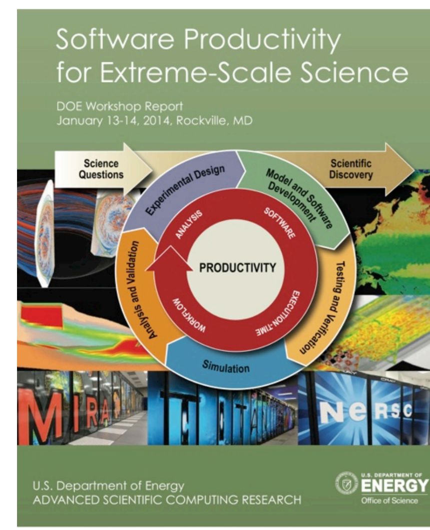
- Productivity costs
  - Human resource in development and re-engineering
  - Machine and energy resources in runtime (*performance*)
  - Utility and correctness of computational results

Exascale Computing Productivity Attention

- DARPA High Productivity Computing Systems
  http://en.wikipedia.org/wiki/High_Productivity_Computing_Systems
- Extreme-Scale Scientific Application Software Productivity: Harnessing the Full Capacity of Extreme-Scale Computing, white paper, September 9, 2013.
  http://www.orau.gov/swproductivity2014/ExtremeScaleScientificApplicationSoftwareProductivity2013.pdf



- Software Productivity for Extreme Scale Science, DOE ASCR Workshop, January 13-14, 2014.
  http://www.orau.gov/swproductivity2014/

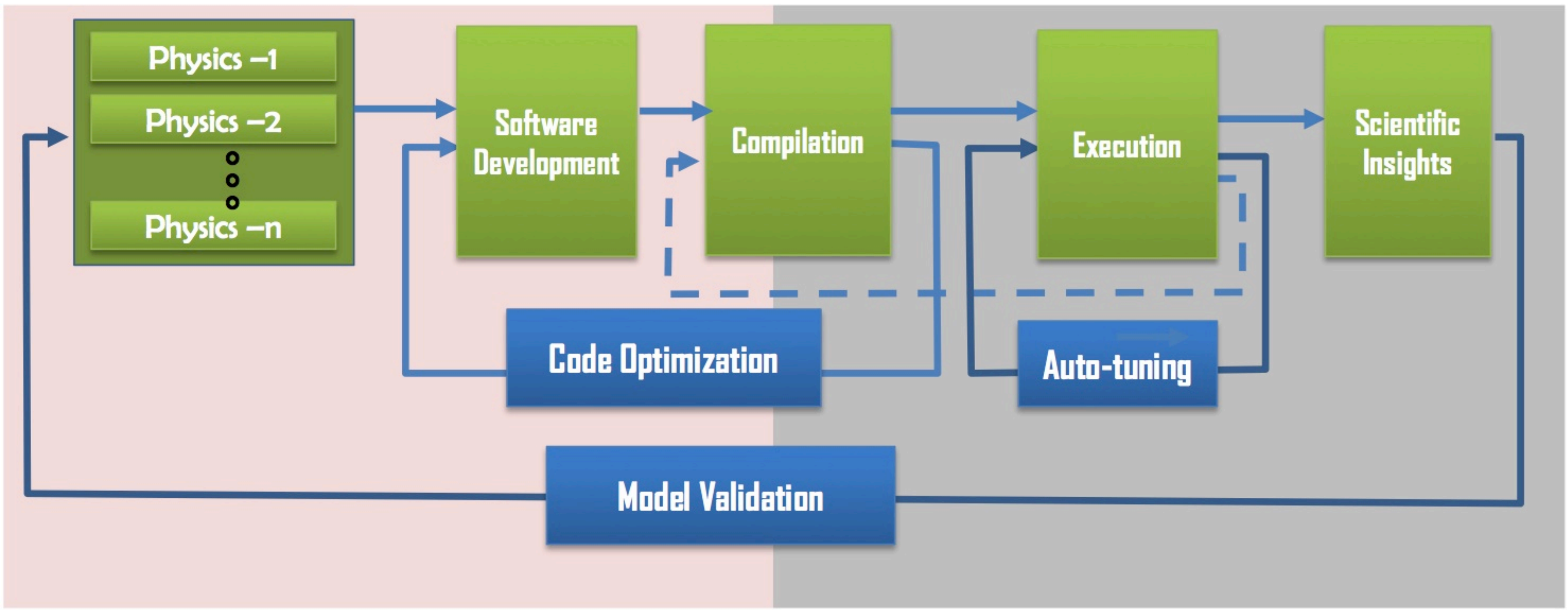


- Exascale Computing Systems Productivity, DOE ASCR Workshop, June 3-4, 2014.
  http://www.orau.gov/ecsproductivity2014/
- ACS Productivity Workshop, DOE Office of Science, July 2014, Indiana University.

What is Exascale Computing Productivity?

- Exascale computing productivity is the effective and efficient use of all exascale resources (hardware, application software, runtime, people, processes, energy) in the production of new scientific insights
- Goal
  - Productivity awareness embedded in all exascale lifecycle activities from R&D through deployment to operation and production of scientific insights
  - Increase efficiency of overall exascale ecosystem during research and development by identifying, removing, and ameliorate productivity and *performance* bottlenecks

Exascale Productivity End-to-End



Application Software Productivity	**Execution-time productivity**
• Metrics/attributes • Ease of use • Heterogeneous portability • Reusability • Interoperability/maintainability • Hardware capabilities discovery • Software performance engineering	• Metrics/attributes/execution models • Observability /controllability/monitoring • Instrumentation/measurement/ traceability • Productivity modeling/simulation: prediction, diagnosis • Performance engineering • Dynamic performance adaptation

Courtesy of Thomas Ndousse-Fetter, DOE

Scientific workflows

Future of Scientific Workflows

- DOE NGNS/CS Scientific Workflows Workshop
  - April 20-21, 2015, Rockville, Maryland

    http://extremescaleresearch.labworks.org/events/workshop-future-scientific-workflows
  - Co-organizers: Ewa Deelman (USC) and Tom Peterka (ANL)
- Workflows for DOE science, energy, security missions
  - Current state-of-the-art (HPC and distributed)
  - Workflow technologies
    - creation, execution, provenance, usability, reproducibility, automation
  - Impact of emerging extreme-scale systems
- Focus on requirements for workflow methods and tools
- Consideration for extreme-scale drivers
  - Application requirements (computational, productivity)
  - Extreme-scale computing technologies and impact on workflow

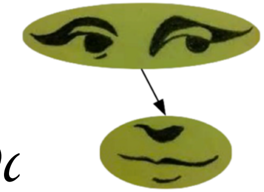
HPC Scientific Workflows

- Current "workflow" for most application scientists:
  - Run a large simulation (maybe performance measurement)
  - Write out a large amount of data
  - Spend a lot of time doing post-processing
  - Repeat (modify experiment or configuration)
- Problem
  - Data analysis requirements are outpacing the performance of parallel file systems
  - Disk-based data management infrastructure limit how often scientists can produce output and the fidelity of analysis
  - Affects scientific insights from simulations
  - Increasing complexity of simulations to drive new knowledge discovery

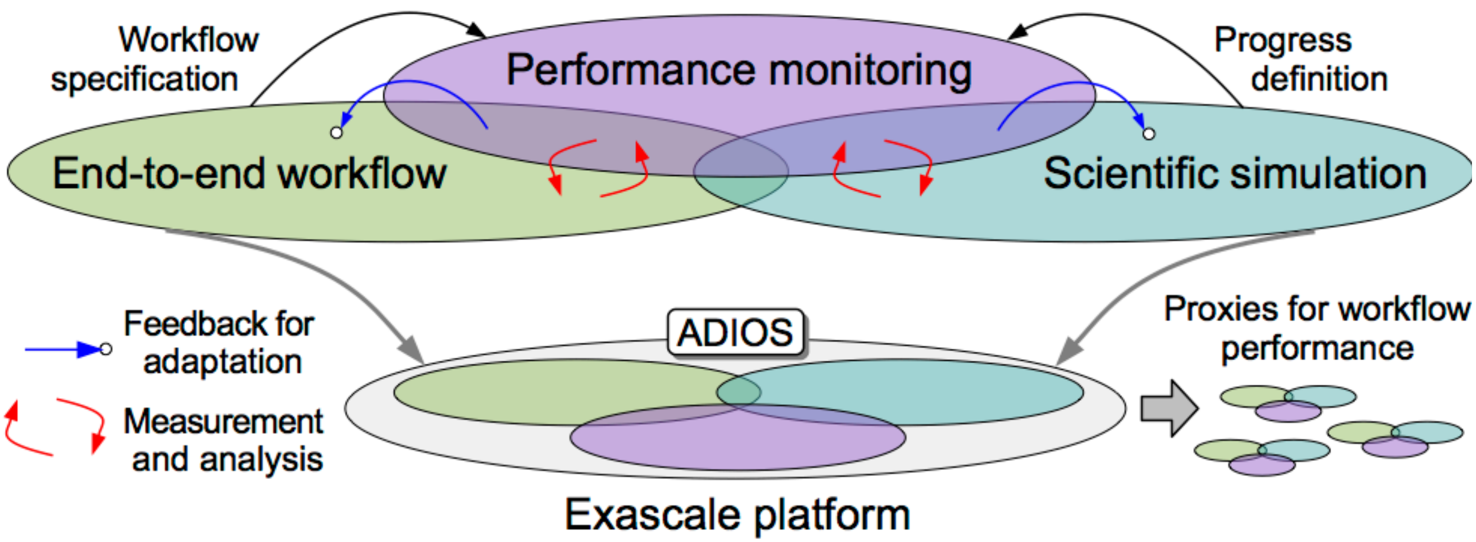
Steps to a Better (Scalable) Workflow

- Try addressing I/O problems with higher-performing data management frameworks
  - ADIOS is being used to abstract I/O (use to create workflow)
  - I/O and data management (flow, staging, …)
- Do as much in situ analytics as possible
  - Run workflow components (analysis, visualization, data management) with computational simulation
    - allow for higher fidelity processing
  - Allocate on dedicated or shared resources
  - Optimize resource usage for in situ scientific workflow
- Requires performance monitoring and analytics
  - Observe workflow (in toto) during execution
  - Use performance information to better configure workflow
  - Possible online workflow resource management

MONA Project



- *Performance Understanding and Analysis for Exascale Da Management Workflows* (MONA) (GT, ORNL, PPPL, UO)
- Explore new methods for performance monitoring and analytics (*monalytics*) of data management actions for exascale simulations
- Data management for end-to-end workflow performance data
  - What performance data to collect (about workflow and components)?
  - How to aggregate, manage, analyze, and visualize data at runtime?
- Create performance models for workflows and workflow proxies
- Co-scheduling of workflow and performance monalytics



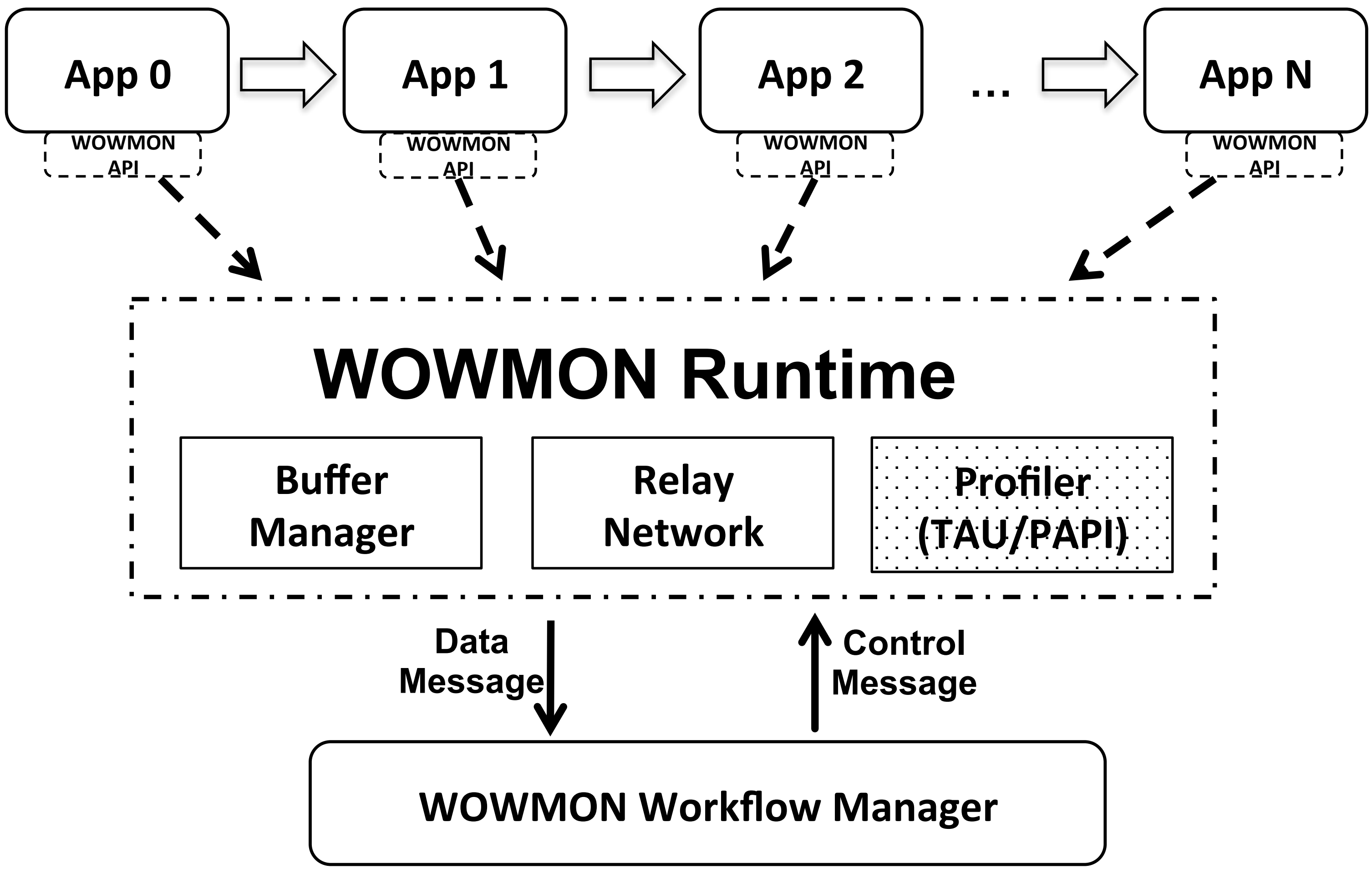
Monalytics

- Need to gain a deeper understanding of where and when performance bottlenecks occur
  - Scientific workflows involve parallel components
  - Properties of scientific workflows (flow)
- Characteristics of monalytics
  - Local operation
    - operate locally and in situ
    - capture aspects of where and when performance data is collected
  - Aggregate performance information
    - measured locally and collected globally
    - modeled as distributed monalytics graphs
    - used specifically for making workflow management decisions
  - Tradeoff of data collection, analysis cost, timeliness
    - Appropriate to what workflow decisions are being made

MONA First Steps

- Create a workflow monitoring (*WOWMON*) infrastructure to capture and analyze information about scientific workflow behavior and performance
- Develop a simple interface for users to instrument codes
  - Workflow component performance (TAU)
  - Workflow component metrics and events (WOWMON API)
- Develop a workflow manager to aggregate and analyze performance data from workflow components
  - Designed with runtime workflow control in mind
  - Very simple prototype
- Develop a lightweight and flexible networking layer (EVPath) for communication of performance data with workflow manager
- Test WOWMON on realistic scientific workflows
- Demonstrate WOWMON with respect to evaluation of end-to-end latency in scientific workflow

WOWMON Architecture



WOWMON API

- Workflow developers need to instrument components using WOWMON APIs

Function	Description
WOWMON_REGISTER_VIEW()	Establish connection to runtime
WOWMON_ADD_EVENTS()	Track hardware/software events
WOWMON_PUT_GLOBAL_DATA()	Notify the networking layer to send data
WOWMON_DEREGISTER_VIEW()	Disconnect from the runtime
WOWMON_INIT_TIMER()	Create a user timer
WOWMON_TRACK_TIMER()	Track a user timer

- The API allows each workflow component to inform the workflow manager of events that occur
- Events contain performance data (metrics defined for a workflow) and metadata
- Monitoring support based on TAUg (global view) model

LAMMPS Scientific Workflow

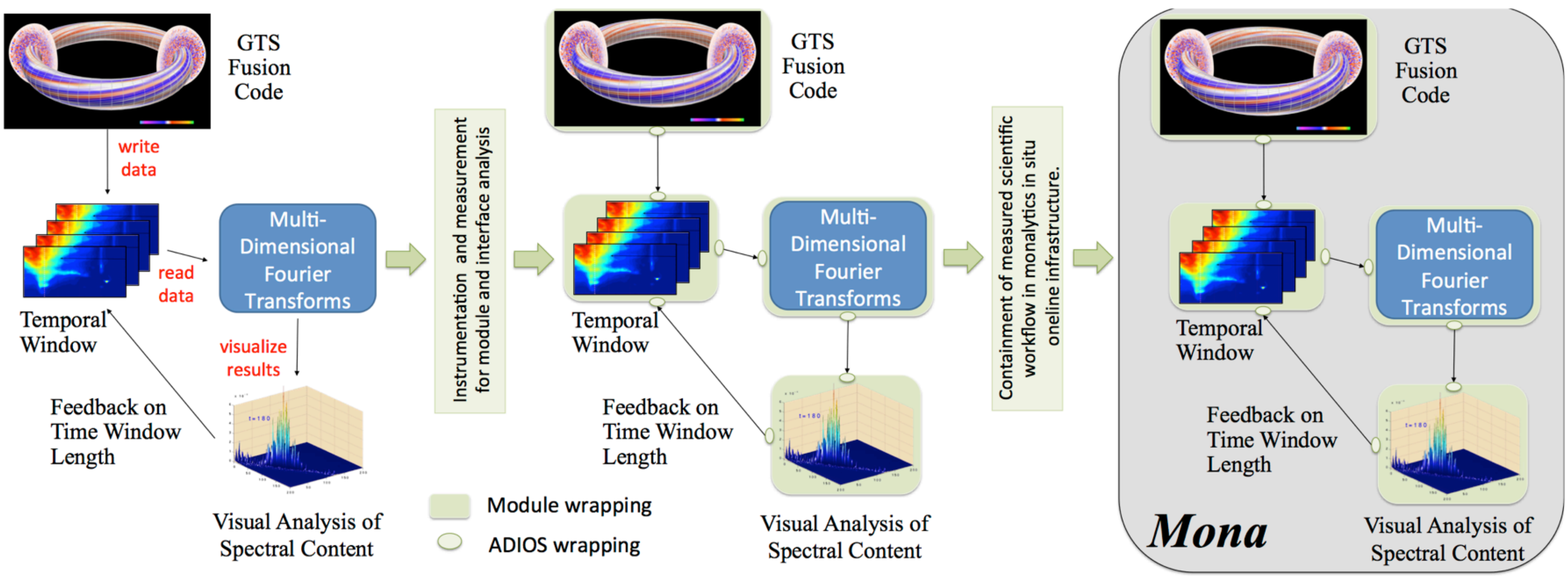
- LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) is a molecular dynamic simulation
  - Extensive set of options for material science study
  - Can be coupled with atomic bond computation (*Bonds*) and symmetry analysis (*Csym*) codes
- Bonds performs all-nearest neighbor calculations to determine which atoms are bonded together
- Csym uses the output of Bonds to further determine whether there is a deformation in the material
  - If deformation is detected, Csym continues to calculate conditions under which a crack occur
  - Potentially feed back this information to LAMMPS
  - Execution time and resource utilization could change

GTS Scientific Workflow

- GTS (Gyrokinetic Tokamak Simulation) is a 3D PIC code for studying effects of turbulence plasma function simulation
  - Coupling of charged particles in plasma and sea waves
- GTS outputs datasets for various purposes
  - Checkpointing, diagnostics, visualization, …
- A GTS scientific workflow utilizes spectral reflectometry analysis (FFT) to determine whether waves grow too fast
  - Indicates a catastrophic disruption of the plasma
  - Important to diagnose in order to provide feedback to simulation
- GTS workflow scaling
  - Output volumes in large-scale production runs are so large that workflow management based on the file system is problematic
  - Investigate in situ workflow implementation with monalytics

GTS Science Driver

- Integrate MONA framework to collect and analyze performance data to understand GTS workflow dynamics and optimize scheduling decisions

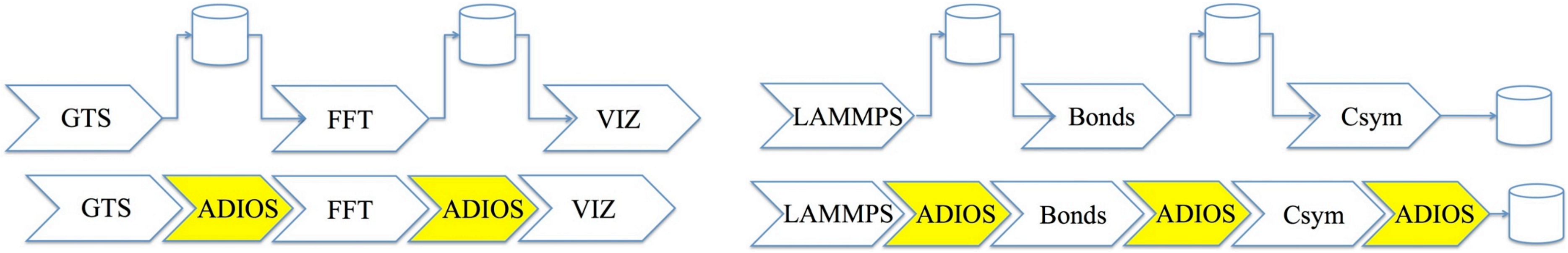


Data-driven Workflow Behavior

- Both LAMMPS and GTS workflows have data-driven behaviors that over time cause varied resource demands
  - Performance variation in workflow components
  - Can affect workflow behavior and delay
- Delays in the workflow can result in back pressure on the simulation progress
  - Could potentially cause slowdown or stalling in simulation
- Production simulation runs can use thousands of cores
  - Small amounts of blocking can waste CPU cycles
- Workflow management attempts to understand the workflow performance factors and make decisions about configuration and resource allocation

ADIOS-enable Scientific Workflow

- Both LAMMPS and GTS scientific workflows have been developed using file-based workflow management
- ADIOS has been used to replace the file I/O and create a memory-based workflow management scheme
  - Memory buffer abstraction for workflow data management
  - ADIOS manages data movement between memory buffers
    - in memory or between nodes
    - supports parallel communication
- Each buffer is managed as a FIFO queue
  - Stores a collection of data objects
  - *Queue Depth* can be configured to adjust queue size



GTS Workflow

LAMMPS Workflow

End-to-End Latency Analysis

- Data-driven behaviors affects execution time of individual workflow components, as well as workflow collective performance (throughput)
- End-to-end latency (EEL) is useful
- How do we measure EEL?
  - Identify key functions which contribute to EEL
  - Capture performance data across the workflow
- Evaluate hardware and software factors that contribute
  - Mapping processes to compute nodes
  - Size of in situ memory buffers
  - Hardware configuration of clusters

End-to-End Latency

- GTS end-to-endlatency
  - Duration from the time that grid data is stored in the memory of the GTS simulation application …
  - … to the time that the visualization output of the FFT is shown on the display
- LAMMPS end-to-end latency
  - Duration from the time that atoms are stored in the memory of the LAMMPS simulation application …
  - … to the time the storage write operation of Csym results
- EEL is determined by both *compute time* of workflow components and *queueing* and *transfer time* as data moves between components

Metrics for LAMMPS Workflow

Metric Name	Function Description
bonds_read_input	Read data and give us a handle on throughput.
bonds_list_output	Output data and give us a handle on throughput out of bonds.
bonds_compute_send	Main computation function in *Bonds*. Most of time is spent in building the adjacent format output.
bonds_read_input_mem	Memory usage for executing bonds_read_input()
bonds_compute_send_mem	Memory usage for executing bonds_compute_send()
csym_read_input	Read data and give us a handle on throughput.
csym_compute_send	Main computation function in *Csym*. Most of time is spent in converting input data.
csym_output_results	Following function csym_output_results() is the end of workflow and where we end latency measurement.
csym_read_input_mem	Memory usage for executing csym_read_input_mem()
csym_compute_send_mem	Memory usage for executing csym_compute_send_mem()
lammps_start_timer	The timer is triggered when generated data is placed in buffer on LAMMPS end.
csym_stop_timer	The timer is triggered when the last analytic finishes.

Metrics for GTS Workflow

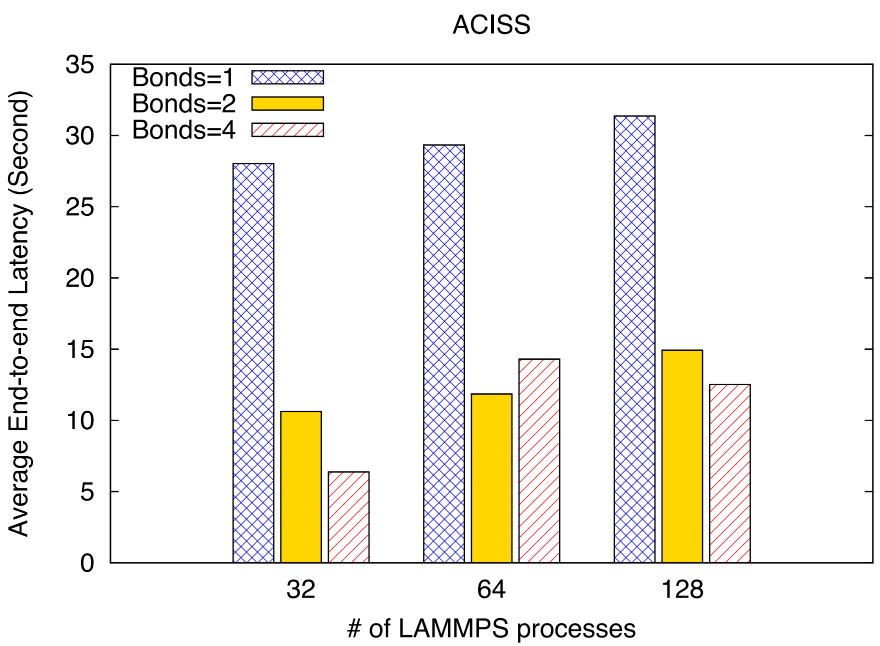
Metric Name	Function Description
gts_restart_write	Output data and give us a handle on throughput out of GTS.
gts_restart_write_mem	Memory usage of executing restart_write()
fft_phi	Main computation function in *FFT*. Most of time is spent on Fourier calculation.
fft_phi_mem	Memory usage for executing fft_phi()
gts_start_timer	The timer is triggered when checkpointing data is placed in networking buffer on GTS end.
fft_stop_timer	The timer is triggered when FFT calculation finishes.

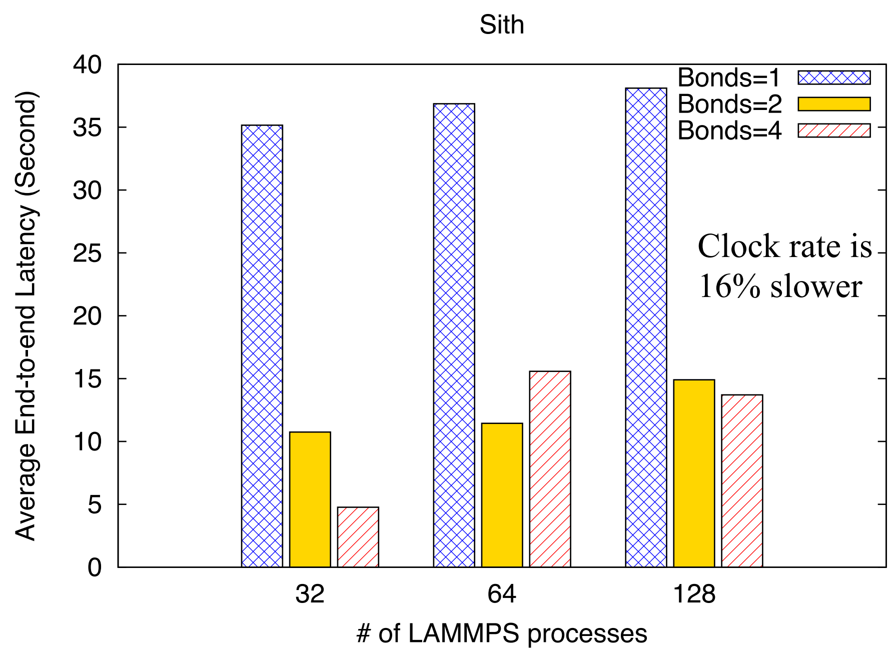
WOWMON Experiments – LAMMPS

- ACISS (UO): 128 12-core Intel Xeon 2.67 GHz, 10GigE
- Sith (ORNLL): 40 32-core AMD Opteron2.3 GHz, IB

```
//LAMMPS: dump_atom.cpp
Void DumpAtom:: write_noimage(int n, double *mybuf)
{
    …
    #ifdef USE_WOWMON
        WOWMON_INIT_TIMER(t, “Start time in LAMMPS”);
        WOWMON_TRACK_TIMER(t);
    #endif
}
```

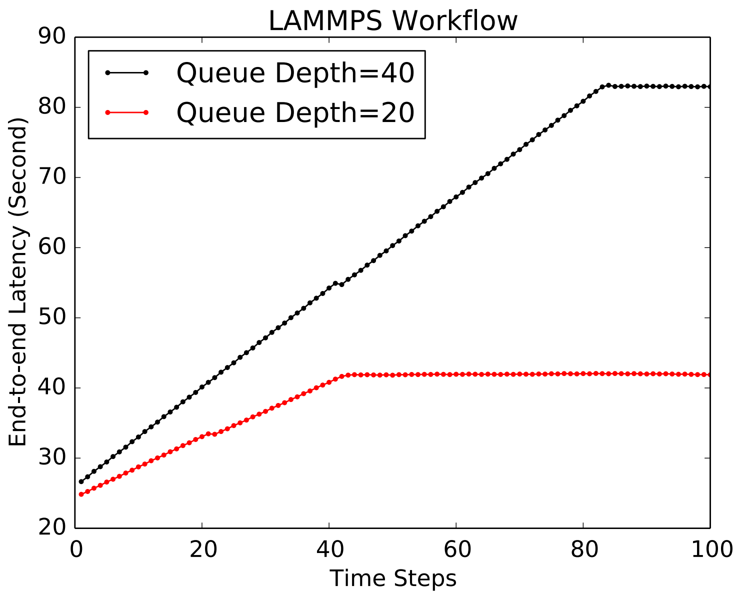
```
//Csym: csym.c
Void compute_and_send(bond_record_ptr atomic_data)
{
    …
    output_results(atomic_adj_data);
    #ifdef USE_WOWMON
        WOWMON_INIT_TIMER(t, ”End time  in csym");
        WOWMON_TRACK_TIMER(t);
    #endif
}
```

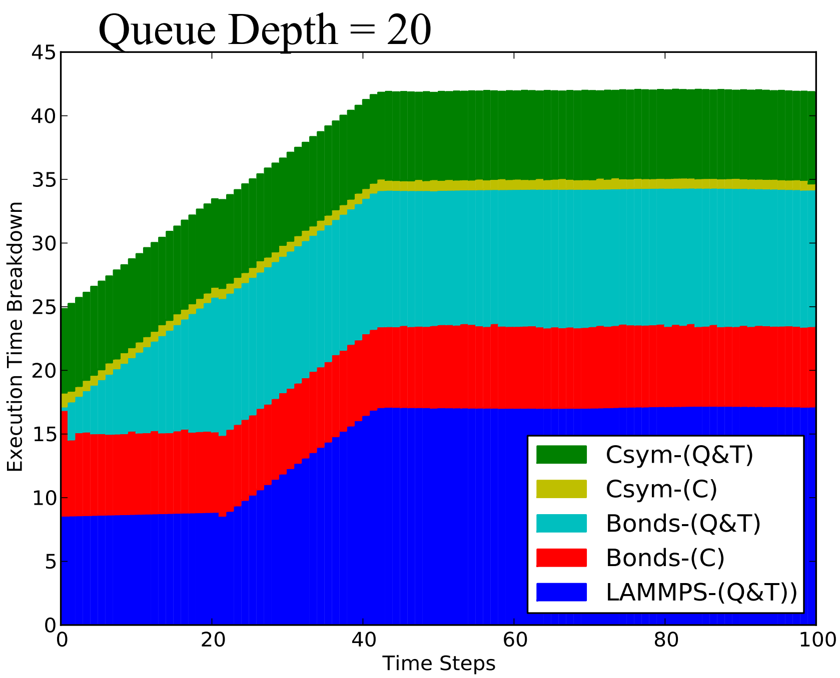


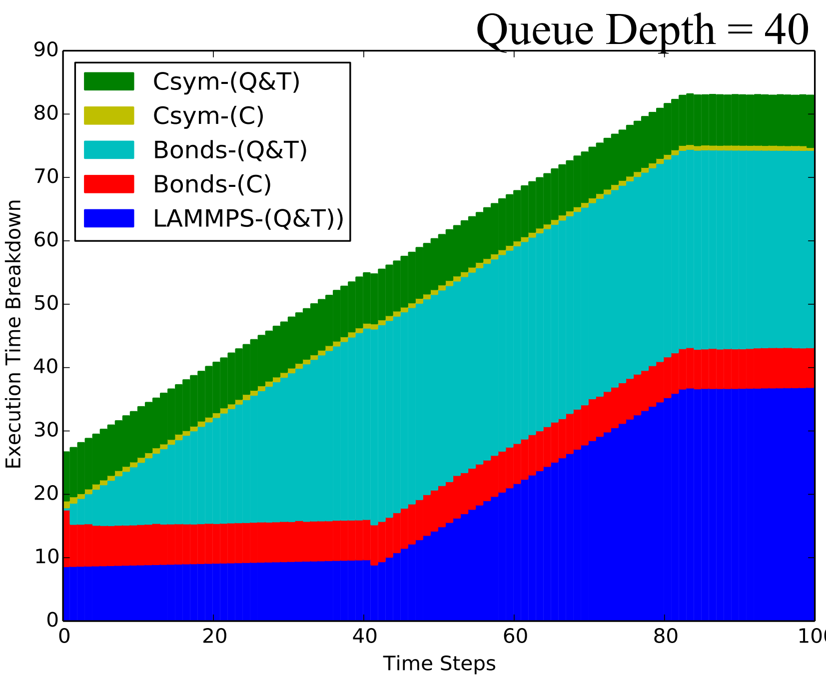


LAMMPS Workflow Performance

- End-to-end latency per time step reflects workflow dynamics
  - Use simple scenario
  - 128 : 1 : 1 process configuration
- Queueing delays cause backup in workflow execution

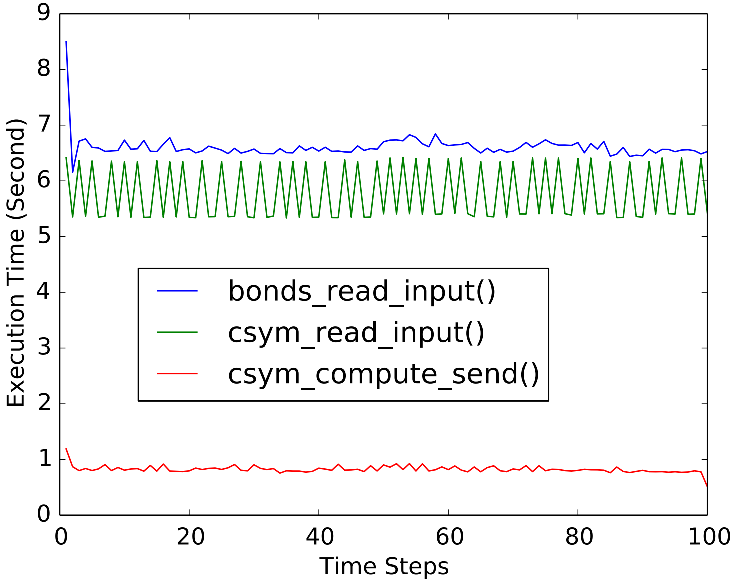




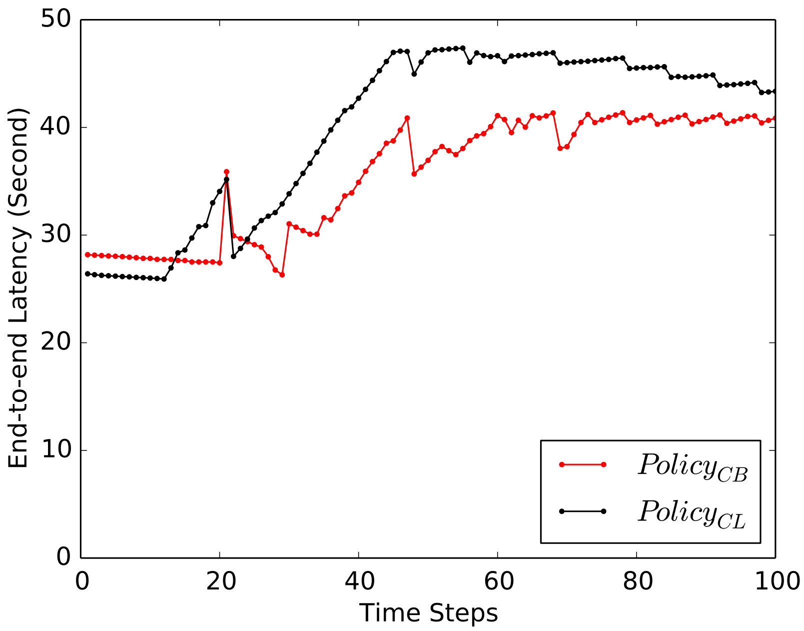


LAMMPS Workflow Performance

- WOWMON collects metrics from the workflow components
  - Allows insight into dynamics of workflow execution



- Can experiment with workload mapping policy to see the effects of process placement
  - $Policy_{CB}$ has *Csym* and *Bonds* placed on a dedicated node with LAMMPS on 10 nodes
  - $Policy_{CL}$ has *Csym* and *LAMMPS* sharing a node

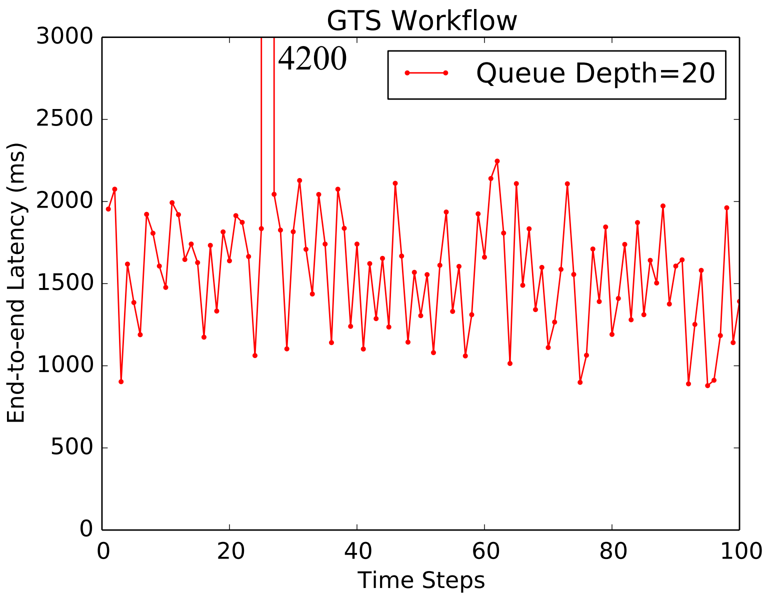


GTS End-to-End Latency

- GTS sends an array *phi* (768 MB per step) to FFT
- Experiments with increasing parallelism (32,64,128)
  - Ratio of parallelism of FFT and GTS is 1:2
- Compare MPI-IO (Atlas Lustre) to in situ (ADIOS)

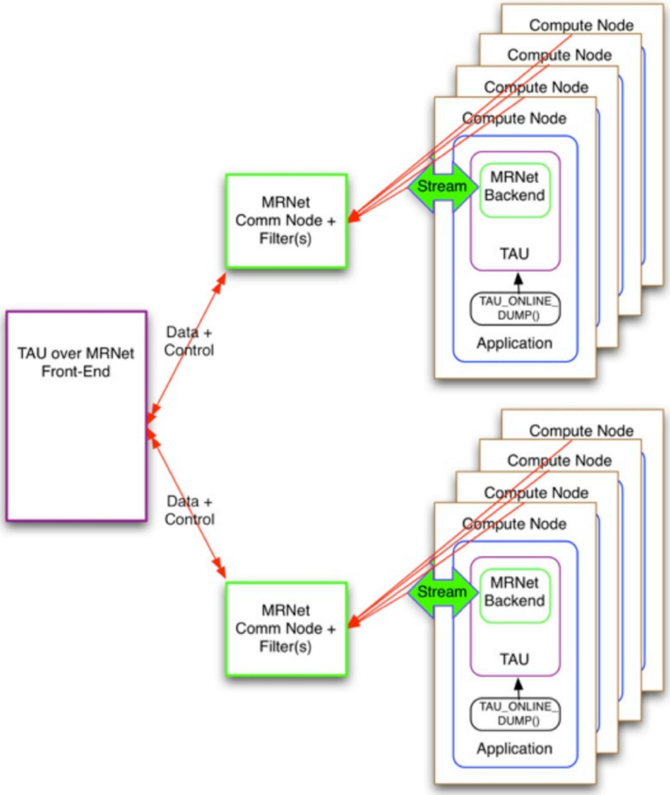
# of Processes	32	64	128
Workflow Execution Time (MPI-IO)	526s	611s	710s
Workflow Execution Time (In-situ)	418s	465s	520s
Average End-to-end Latency (In-situ)	1.33s	1.26s	1.2s

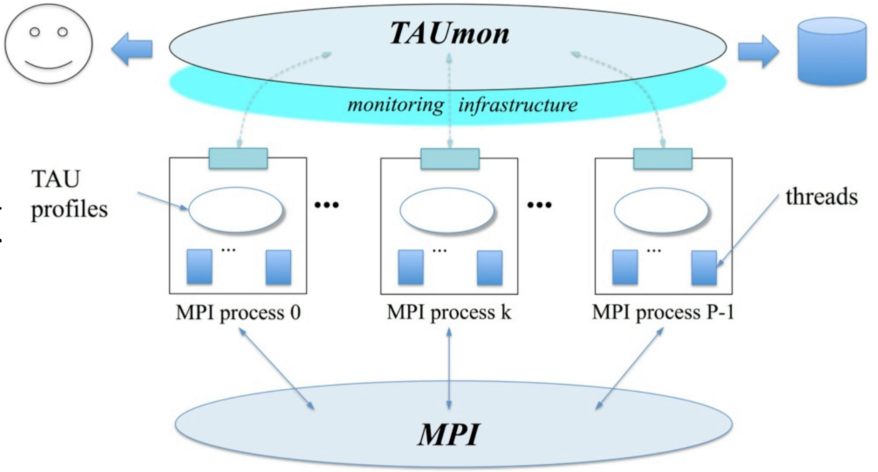
- Look at end-to-end latency on 64 process run on Sith
  - Could be due to GTS computation variation during workflow execution



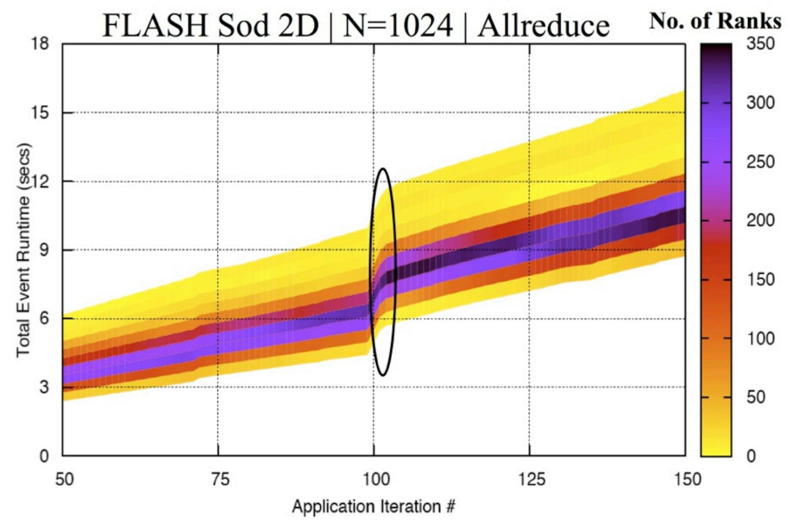
Parallel Performance Monitoring

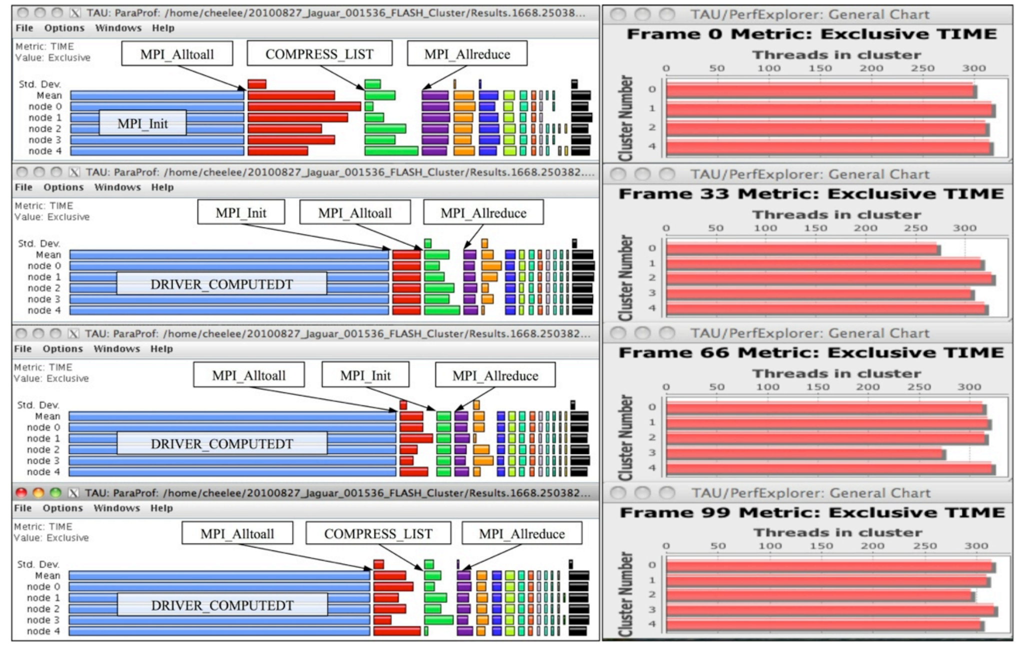
- ***TAUoverSupermon***
  - Supermon monitor from Los Alamos National Laboratory
- ***TAUoverMRNET***
  - MRNet from Wisconsin
- ***TAUg***
  - MPI-based infrastructure to provide global view of TAU profile data
- ***TAUmon***
  - Transport-neutral (SuperMon, MRNet, MPI)
- Develop online analysis metho
  - Aggregation, statistics, …



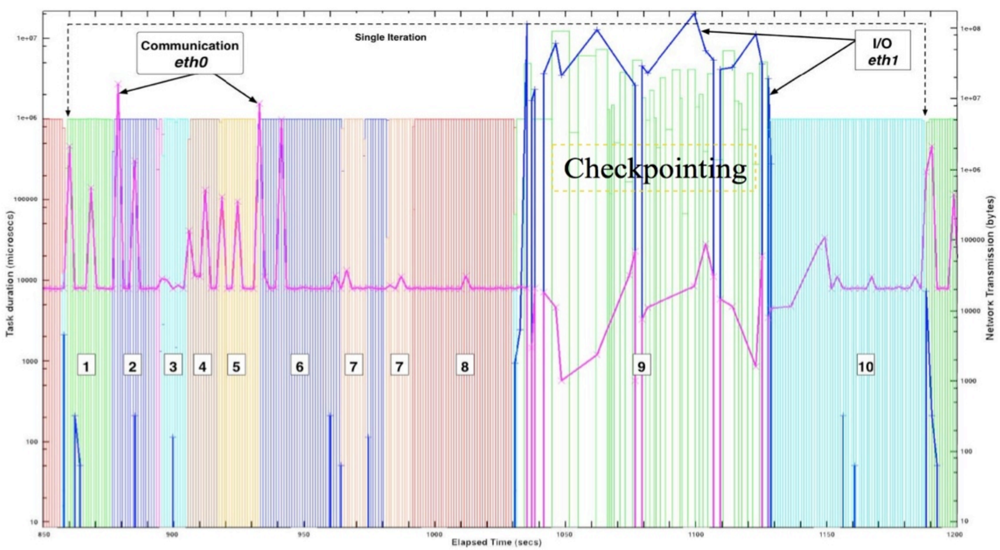


TAUMon Online Analysis

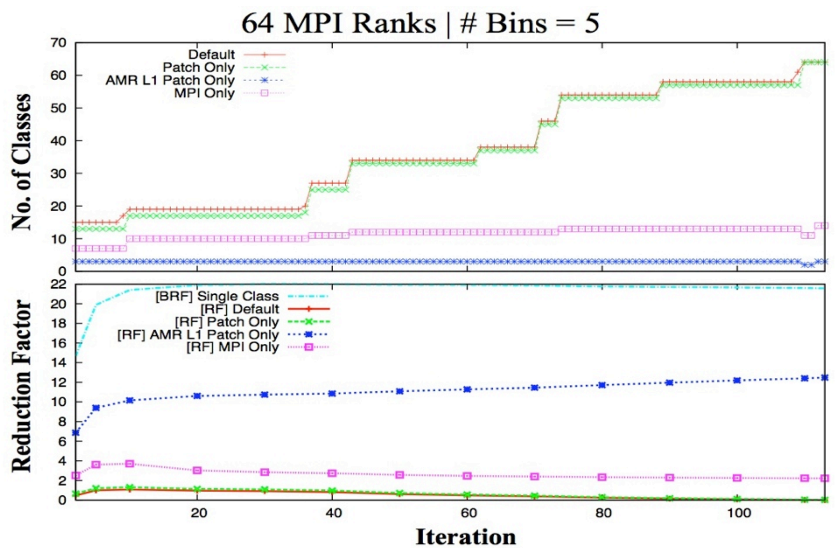




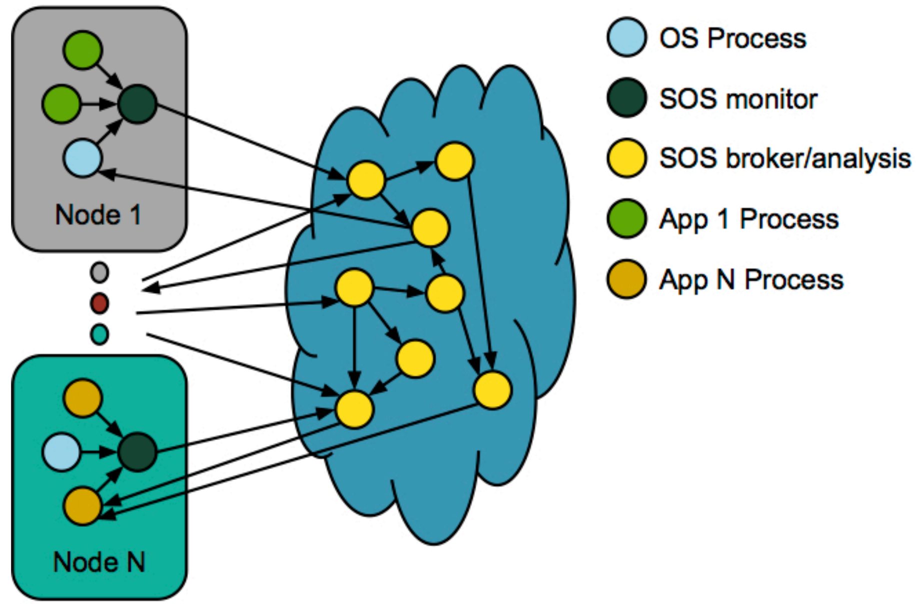
FLASH



Uintah



Scalable Observation System (SOS)



- HPC platform service
  - Single instance, system wide
  - Serves applications, RTS, OS
  - Shares metrics and analysis
  - Framework-like interaction
  - (Possible) dedicated resources
- SOS monitors coordinate node to analysis cloud interactions
  - Any process on node can expose data via SOS
- SOS cloud self organizes to efficiently process data
- Control can go directly to interested processes
- Develop SOS MPMD MPI prototype
  - Running on BG/Q, Cray, and Linux platforms
  - Use for testing and experimentation

Summary

- Scientific workflows take different forms
- LAMMPS and GTS workflows couple HPC simulations with in situ data management, analytics, visualization
- There are workflows that run multiple experiments for exploration, parameter sweep studies, uncertainty quantification, and many other purposes.
- There are many-task computing (MTC) workflows
  - Integrated Plasma Simulator (IPS) framework
- There are data flow programming systems like Swift that have been extended to MTC environments (Swift/T)
- Workflow performance monitoring and analytics will have different objectives for different workflow systems
- WOWMON is a prototype being used to gain experience

Future of Scientific Workflows (2)

- Research areas
  - Systems design and execution
    - scalability, control, data flow, management, monitoring
  - Programming and usability
    - programming models, design patterns, user interface, portability
  - Provenance
    - data/metadata capture, communication, storage, data mining
  - Validation
    - comparing workflow performance to model predictions
    - comparing science results
    - reproducibility of workflow across environments
  - Workflow science
    - formalism of theories, models, environments

Future of Scientific Workflows (3)

- Better understand science processes in simulations, experiments, and collaborations to design workflow management systems (WMS)
- Better understand impact of extreme-scale architectures on WMS design and technology opportunities to support workflow execution
- Better understand the intersection of workflow systems and system software in resource management and scheduling
- Research needed in the WMS design of control and data flows, data models, and programming interfaces
- Better capture of provenance information during and after workflow execution for validating performance and correctness
- Need benchmarks and community data sets for workflow research
- Workflow science that systematically studies the theory, modeling, and benchmarking of workflows