

# VERIFI Workshop

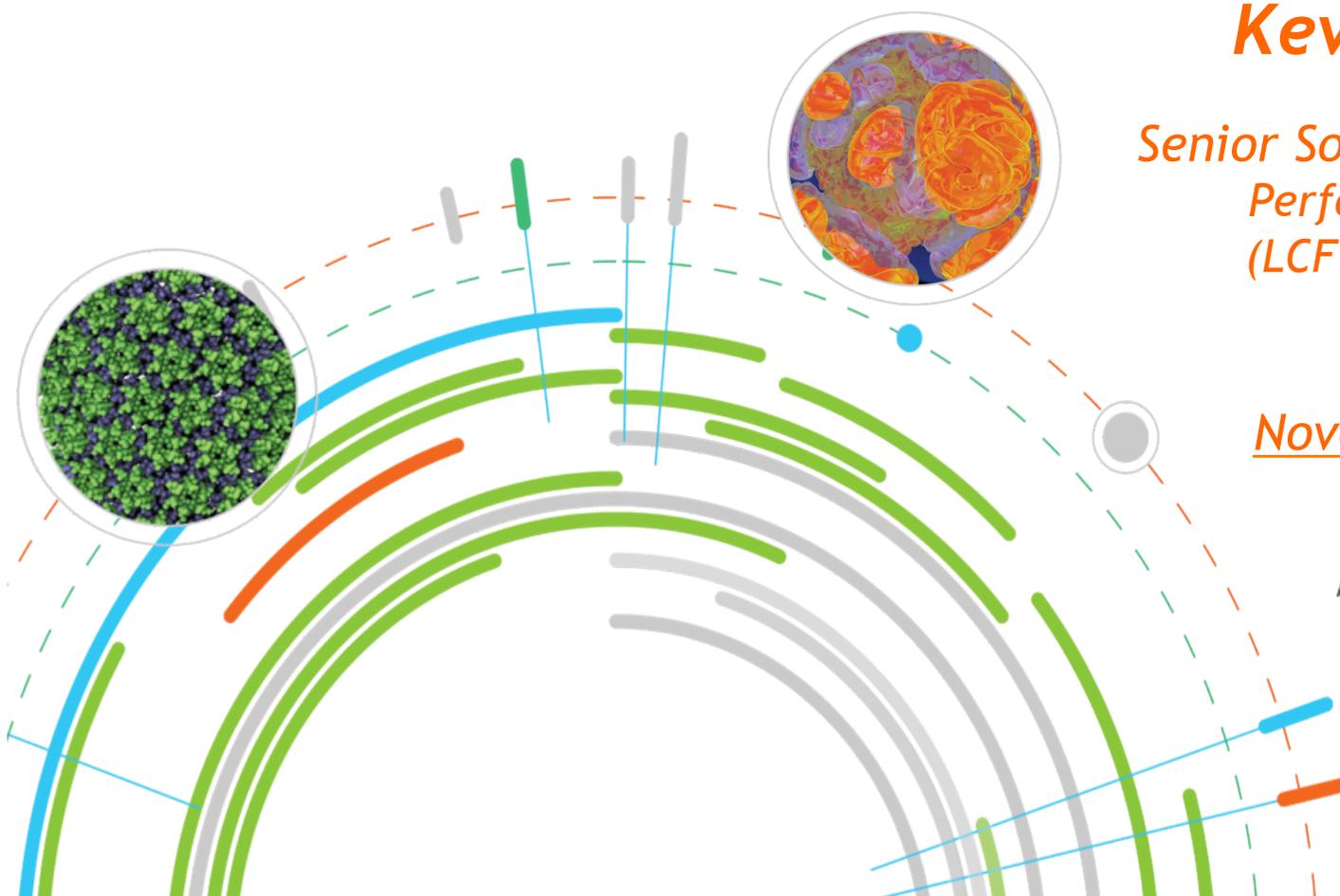
## Scaling up CONVERGE™: ALCF code improvements

*Kevin Harms*

*Senior Software Developer  
Performance Team  
(LCF Division, ANL)*

*November 13<sup>th</sup>, 2014*

Argonne Leadership  
Computing Facility



# Performance Tuning

- ⊙ Argonne

- ⊙ Kevin Harms (LCF)
- ⊙ Marta García (LCF)
- ⊙ Janardhan Kodavasal (ES/LCF)
- ⊙ Sibendu Som (ES)
- ⊙ Yuanjiang Pei (ES)

- ⊙ Convergent Science, Inc. (CSI)

- ⊙ Priyesh Srivastava
- ⊙ Shaoping Quan
- ⊙ Keith Richards



# Disclaimer

- ⊙ Work considered “pre-production”, *Beta* version
- ⊙ CSI has been involved with code changes and performance optimizations
- ⊙ Not all code necessarily the final implementation



# Performance Tuning

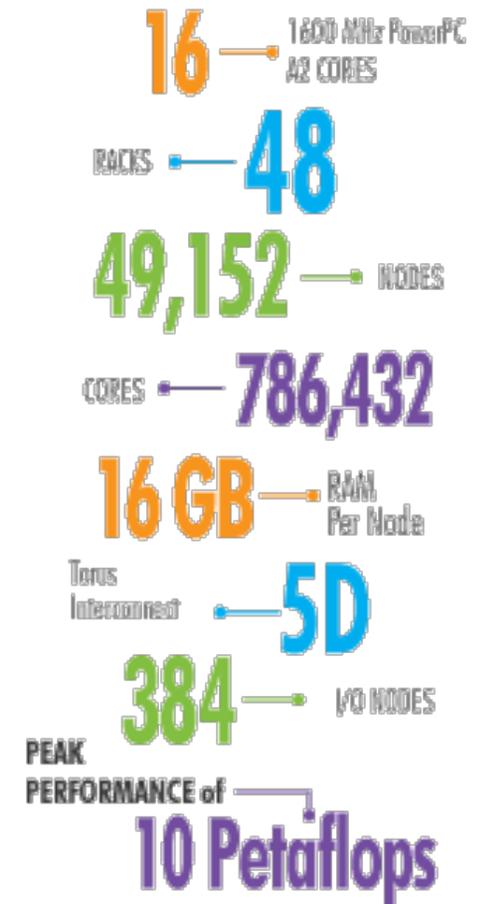
- ⊙ New compute, I/O, system architectures bring new challenges
- ⊙ Common “problem” when bringing new codes to Mira
  - ⊙ ALCF staff available to help address the issues
- ⊙ Typical issues
  - ⊙ Scaling
  - ⊙ I/O
  - ⊙ Microarchitecture
  - ⊙ OS
- ⊙ Based on CONVERGE version 2.1

Computational bottlenecks typically  
show up at scale...

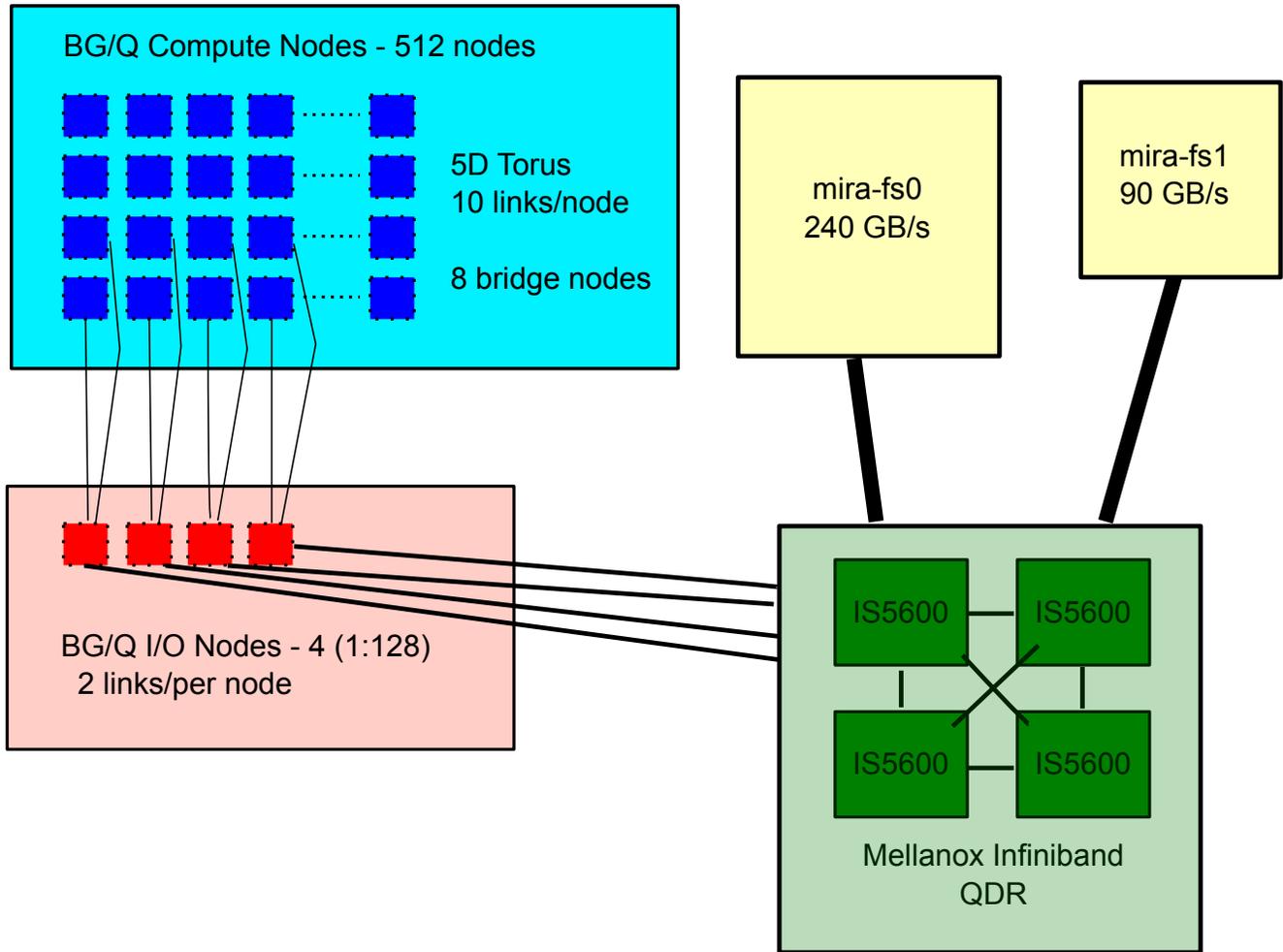


# Background on Mira

- ◎ Compute Architecture
  - ◎ PowerPC A2 - 16 cores - 1.6 GHz
  - ◎ 49,152 compute nodes!
  - ◎ Xeon nodes to Mira node
    - Mira - 204 GFLOP - 1/8<sup>th</sup> for Integers
    - Dual Xeon E5 2650 16 cores @ 2.0GHz AVX - 262 GFLOP
    - Instruction Mix for CONVERGE: 8% FP / 92% INT
- ◎ I/O Architecture
  - ◎ I/O Forwarding Nodes (IONs)
    - 128 Compute nodes to 1 I/O node
    - Decreases complexity for file system (fewer clients)
    - Implies some limitations on I/O design
  - ◎ Parallel File System - IBM GPFS
    - mira-fs1 - 90 GB/s
- ◎ Network Architecture
  - ◎ 5D Torus, Hardware optimization for collectives
  - ◎ Message Rate @ 49k - 11.91 Mm/s
  - ◎ BW @ 48k - 21,827.74 GB/s

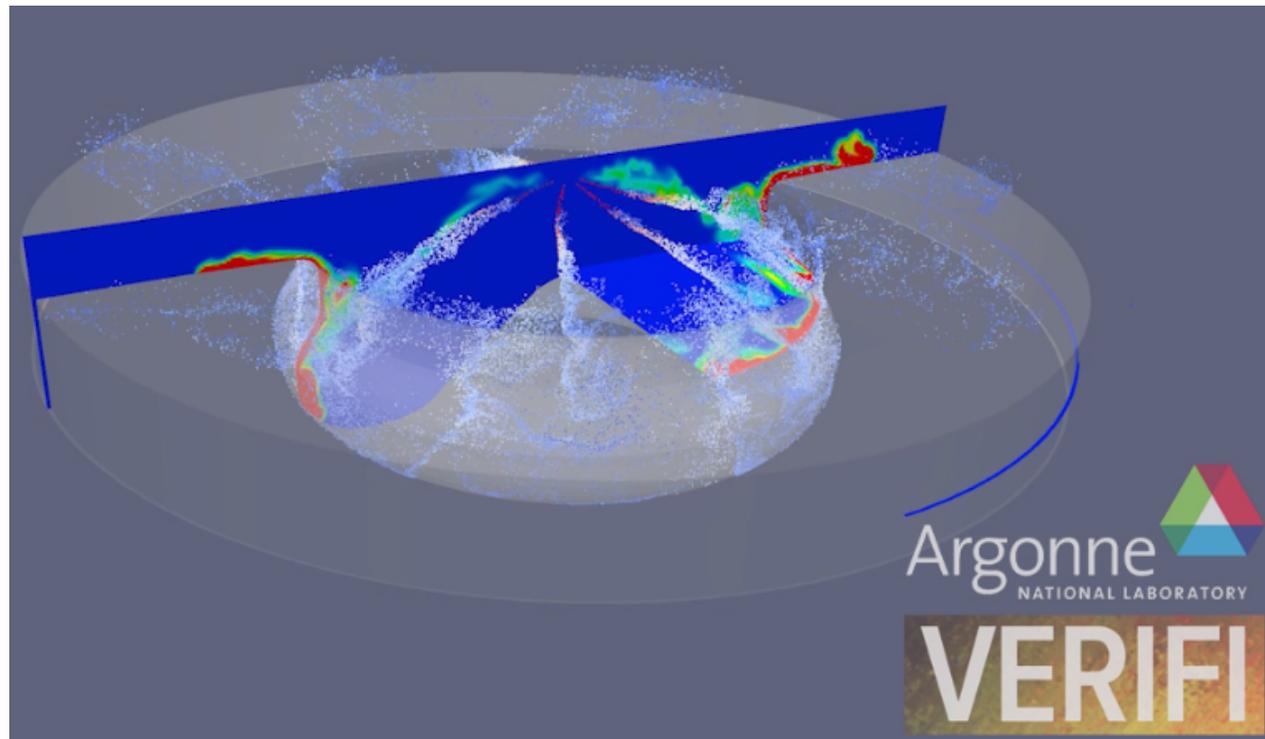


# Mira Overview



# Case used for tuning - GCI hands-on

- ⦿ 360 deg. Gasoline Compression Ignition VERIFI hands-on case
  - ⦿ 128 nodes, 16 cores/node ---> **2048 cores**
  - ⦿ Closed cycle, LES, 0.15 mm uniform mesh (no AMR/embedding)
  - ⦿ Chemistry solved in every cell
  - ⦿ ~25 million cells near injection, ~9 million cells near TDC/ignition
  - ⦿ Focus was to understand I/O issues and load-imbalance **near TDC/ignition**



# Analysis Tools

- ⦿ Rice HPC Toolkit - sample based profiling
- ⦿ TAU - tracing
- ⦿ HPM - hardware counter performance monitor
- ⦿ Darshan - I/O statistics

# I/O Tuning - Restarting a simulation (read)

- ⊙ **Restarting on 2048 cores**

- ⊙ Restart approximately 4-5 GB near TDC
- ⊙ Each MPI rank loads the restart data (2048 ranks in all)

- ⊙ **Issues**

- ⊙ All ranks try to load restart in parallel, results in lock contention within GPFS
- ⊙ Contention of resources at I/O node
- ⊙ Many system calls used -> high overhead for these due to forwarding

- ⊙ **Resolutions**

- ⊙ Single rank reads restart data and broadcasts to all ranks
- ⊙ Reduce system call overhead with caching data locally at compute node

- ⊙ **Results**

- ⊙ Previous: >2 hours
- ⊙ Current: ~5 minutes

**Solution: Only 1 rank reads**



# I/O Tuning - Writing Restart Files (write)

- ⦿ **Writing the restart file**
  - ⦿ Restart about 4-5 GB near TDC
  - ⦿ Each MPI rank writes output to restart in serial
- ⦿ **Issues**
  - ⦿ Each MPI rank writes its contribution to the restart in serial
    - Serial solution becomes slow as simulation is scaled to 1000s of processes
  - ⦿ Significant number of system calls have a high overhead on BG I/O
- ⦿ **Resolutions**
  - ⦿ Restart written in parallel
    - Each MPI rank accumulates its changes into a memory buffer
    - MPI-IO interfaces for parallel collective write
  - ⦿ In memory buffering reduces system calls
- ⦿ **Results**
  - ⦿ Previous: 1563 seconds
  - ⦿ Current: 2-2 seconds (**700x speedup**)

Solution: Parallel writes

# I/O Tuning - Writing Post-processing files

- ⦿ **Writing the post processing files**

- ⦿ Post file about 1 GB
- ⦿ Each MPI rank writes output to post file in serial

- ⦿ **Issues**

- ⦿ Each MPI rank writes its contribution to the post file in serial
  - Serial solution becomes slow as simulation is scaled to 1000s of processes
- ⦿ Significant number of system calls have a high overhead cost on BG I/O architecture

- ⦿ **Resolutions**

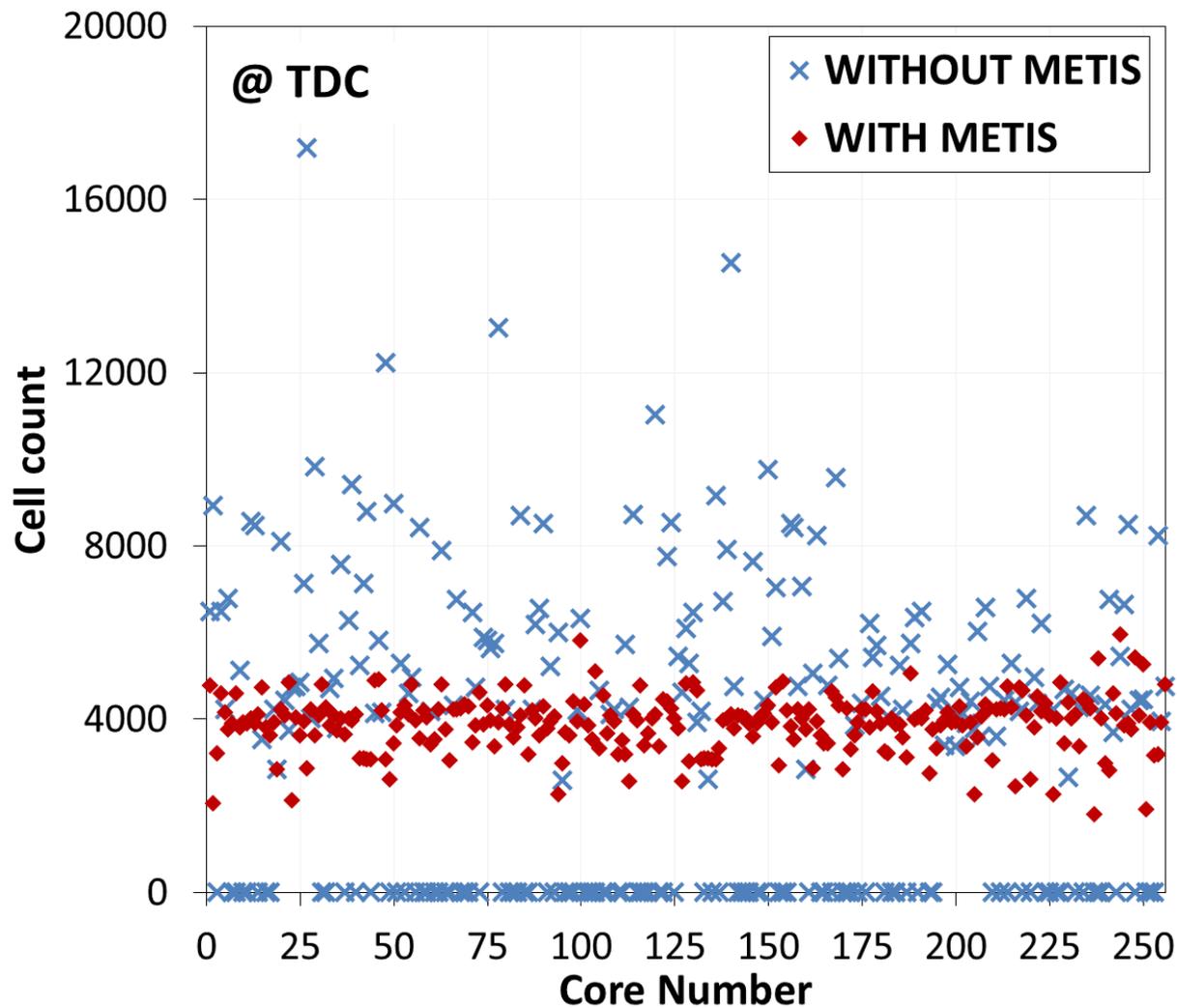
- ⦿ Post file written in parallel
  - Each MPI rank accumulates its changes into a memory buffer
  - MPI-IO interfaces for parallel collective write
- ⦿ In memory buffering reduces system calls

- ⦿ **Results**

- ⦿ Previous: 645 seconds
- ⦿ Current: 1-2 seconds (**600x speedup**)

Solution: Parallel writes

# Fluid mechanics Load Balance - Previous work



ALCF/VERIFI  
modifications in  
2013 improved  
load distribution  
with METIS



Chemistry load  
imbalance was still  
an issue

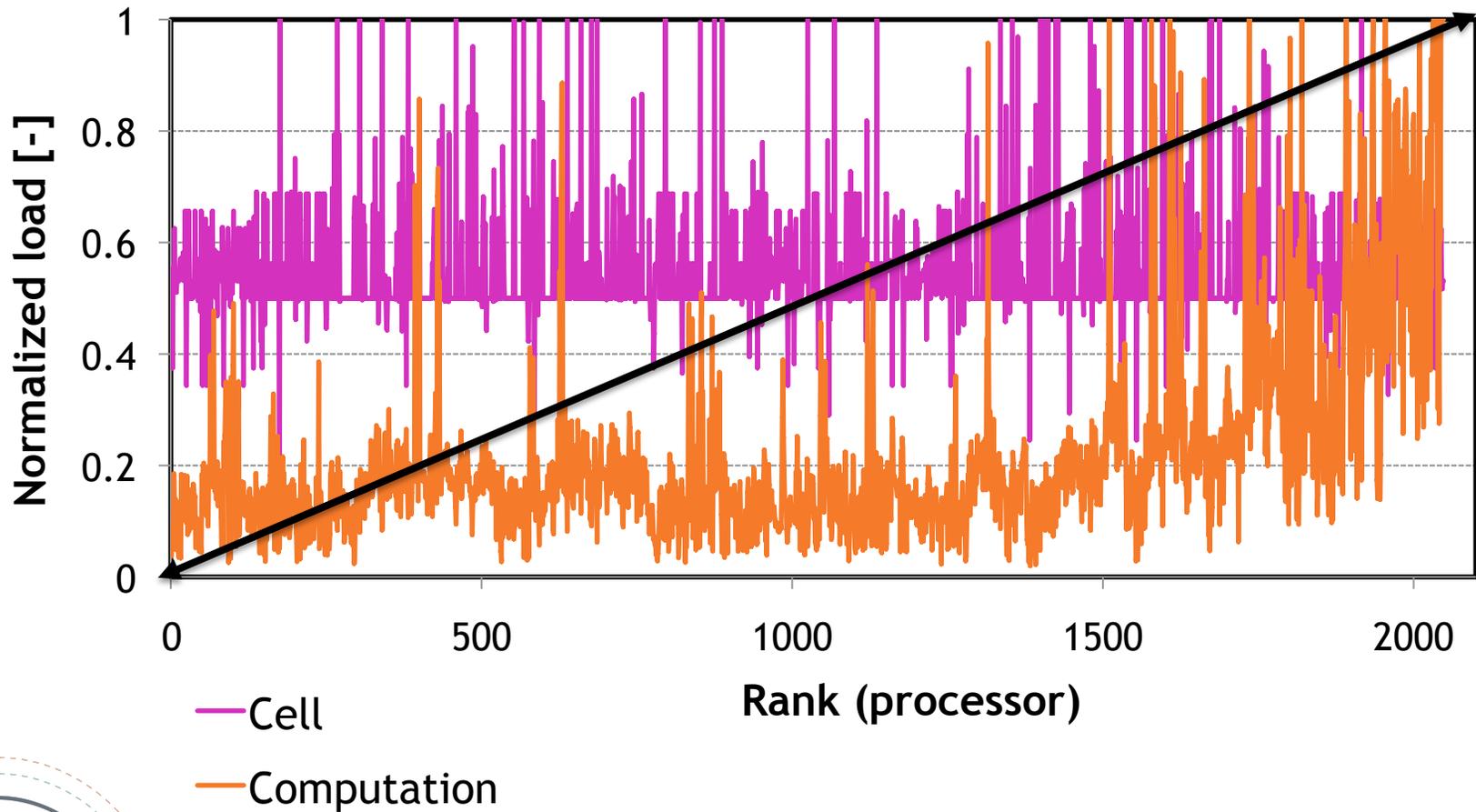
# Load Balancing of Chemistry calculations

- ⊙ **Chemistry calculations were balanced on a cell basis in orig. code**
  - ⊙ Cells are distributed evenly over MPI ranks
- ⊙ **Issues**
  - ⊙ Each cell does not constitute an equal amount of computational work
  - ⊙ Some cells have “stiffer” chemistry going on....
- ⊙ **Resolutions**
  - ⊙ Weight cells by stiffness for balance
  - ⊙ Balance adjusted as simulation proceeds
  - ⊙ Implemented into code by CSI based on VERIFI/ALCF recommendations
- ⊙ **Results** - 24° SOI case for a 3hr simulation window near ignition
  - ⊙ Previous: 42 timesteps
  - ⊙ Current: 143 timesteps (3.4x speedup)

**Solution:**  
Distribute chemistry calls based on stiffness

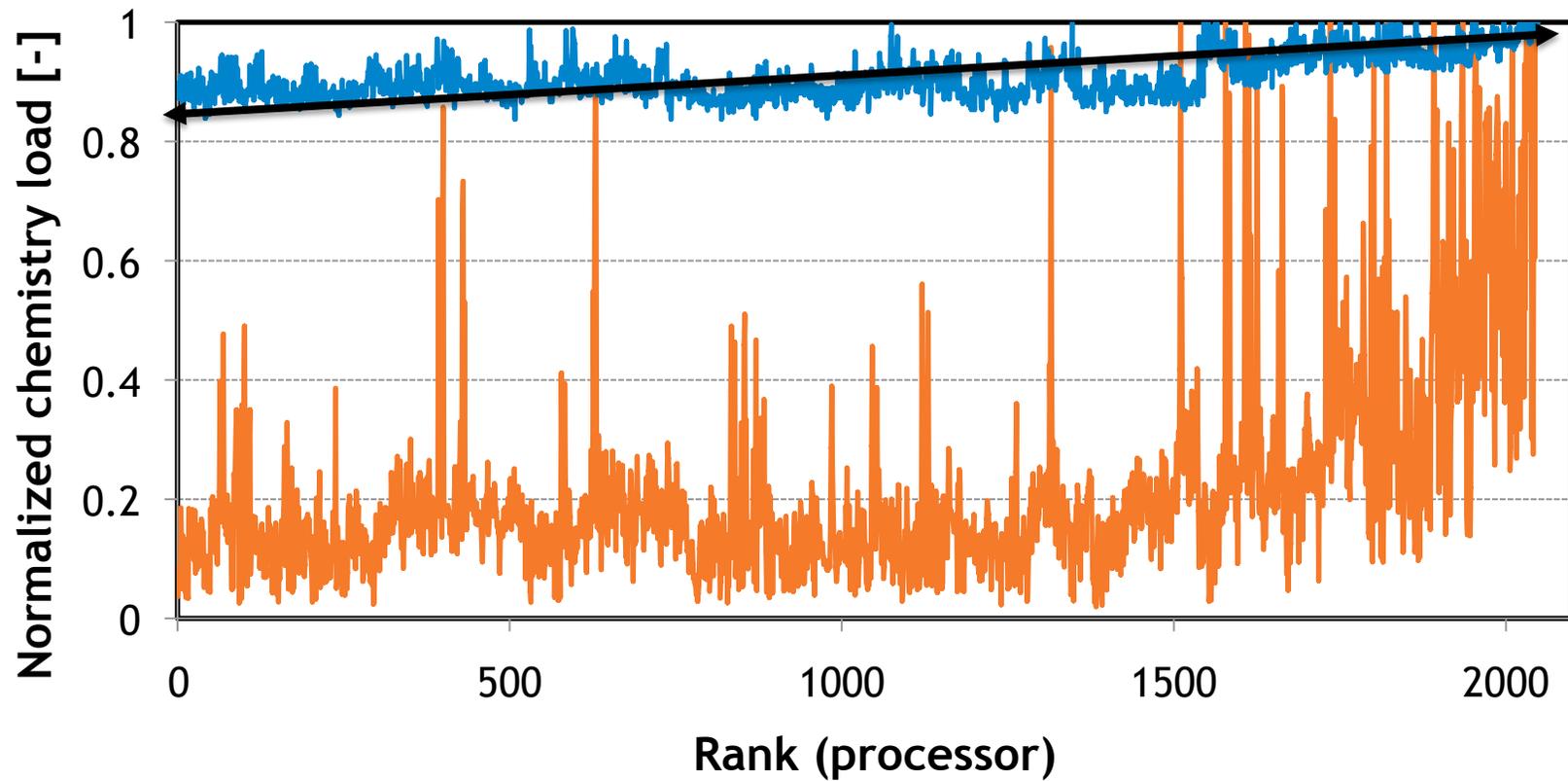
# Chemistry Load Balance - Initial

>10X computation load imbalance



# Chemistry Load Balance - Improved

<1.2X imbalance with stiffness-based load balance



— Computation - orig    — Computation - new



# Threading

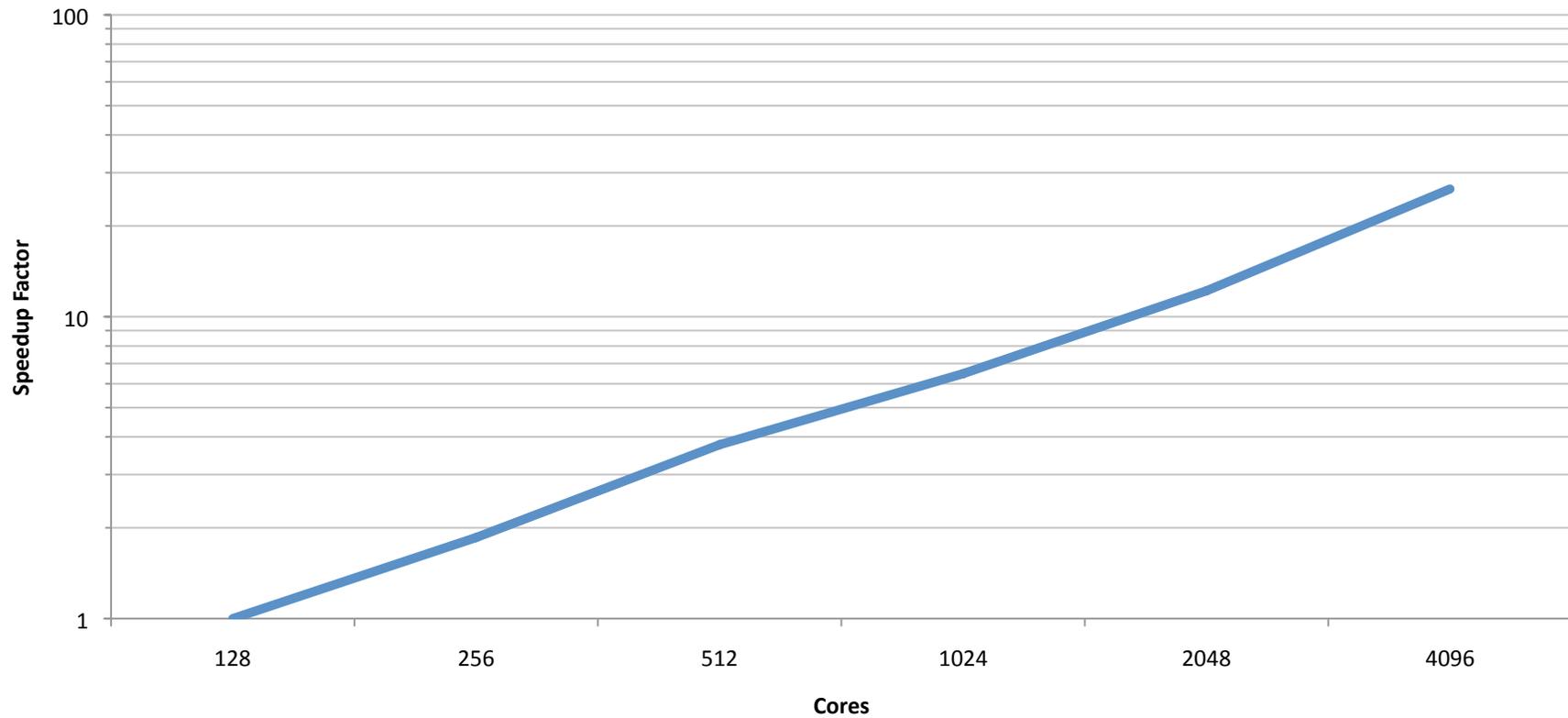
- ⊙ **Computational sections are single threaded**
- ⊙ **Issues**
  - ⊙ BG architecture supports issuing 2 instruction per cycle
    - only 1 instruction can be issued from a single hardware thread
    - Need to run at least 2 hardware threads to maintain peak issue rate
  - ⊙ Modern Intel chips support issuing 2 instructions from 1 HW thread
- ⊙ **Resolutions**
  - ⊙ Add OpenMP annotation to critical compute routines
  - ⊙ Run with 4 hardware threads
- ⊙ **Results**
  - ⊙ 1 HW Thread: 2080 seconds
  - ⊙ 4 HW Thread: 1972 seconds (5% speedup)

Do OpenMP/threading where possible



# Scaling Chart

Scaling Performance (compute time)  
SOI -35° aTDC [0.66° - 0.74° CA]

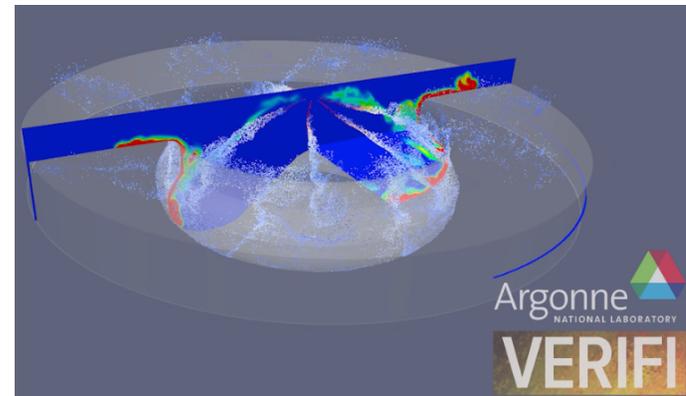


# Summary of ALCF/VERIFI improvements

- ⊙ Output (writing post\* and restart) : **700x**
- ⊙ Input (reading restarts) : **>20x**
- ⊙ Chemistry load balance : **8x** (3.4x speedup in simulation)
- ⊙ Speedup from multithreading : **1.05x**

# Future Work

- ⊙ Scaling to 16k processors (1 rack of Mira)
  - ⊙ Communication algorithm improvements
  - ⊙ Memory optimization
  - ⊙ 250 million cells (~15k cells per process) / 1 billion cells for ensemble
- ⊙ Improve compute performance
  - ⊙ Find opportunities to vectorize loops
  - ⊙ Improve Floating Point instruction mix
  - ⊙ Improve load balance when AMR is used
- ⊙ Standardize on little endian file format for restarts
  - ⊙ Allow data sets to move seamlessly between BG and traditional clusters



# Questions

- ◉ [wiki.alcf.anl.gov](http://wiki.alcf.anl.gov)
- ◉ [support@alcf.anl.gov](mailto:support@alcf.anl.gov)
- ◉ This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under contract DE-AC02-06CH11357.

