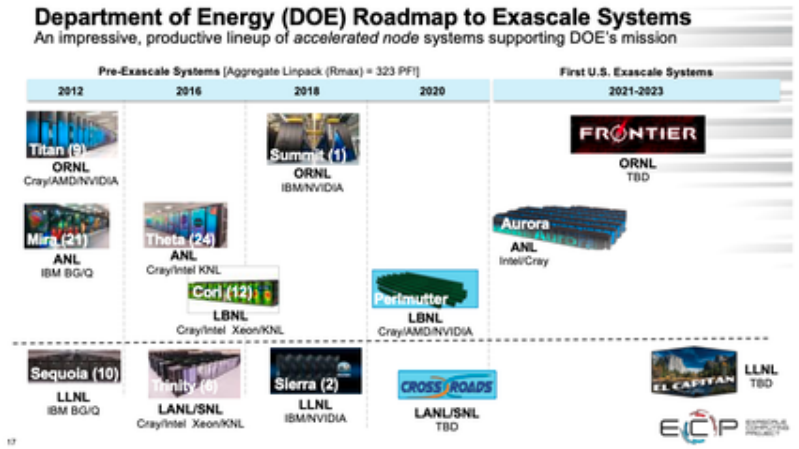
Enhancing the Performance and Scalability of Third-Party Libraries in the Sandia I/O Software Stack

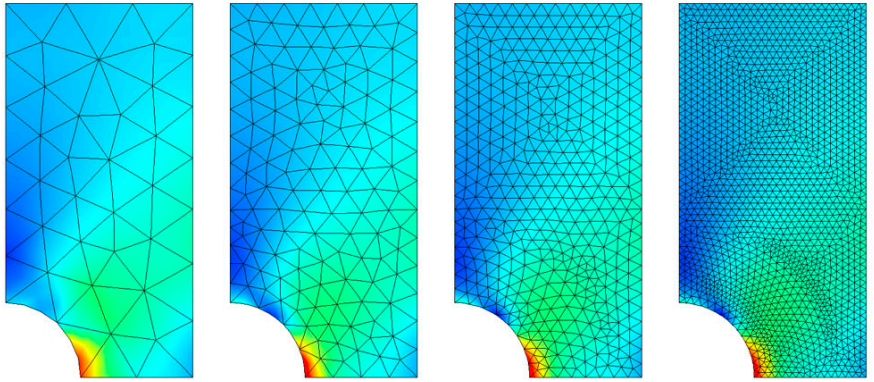
COMPSIM

HDF5 User Group Meeting 2020, October 14-16, 2020

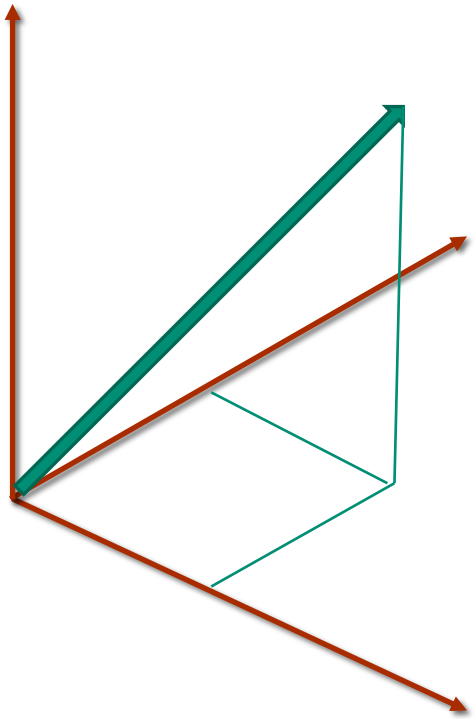
Gregory Sjaardema, Engineering Sciences Center,
Sandia National Laboratories, Albuquerque NM

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.



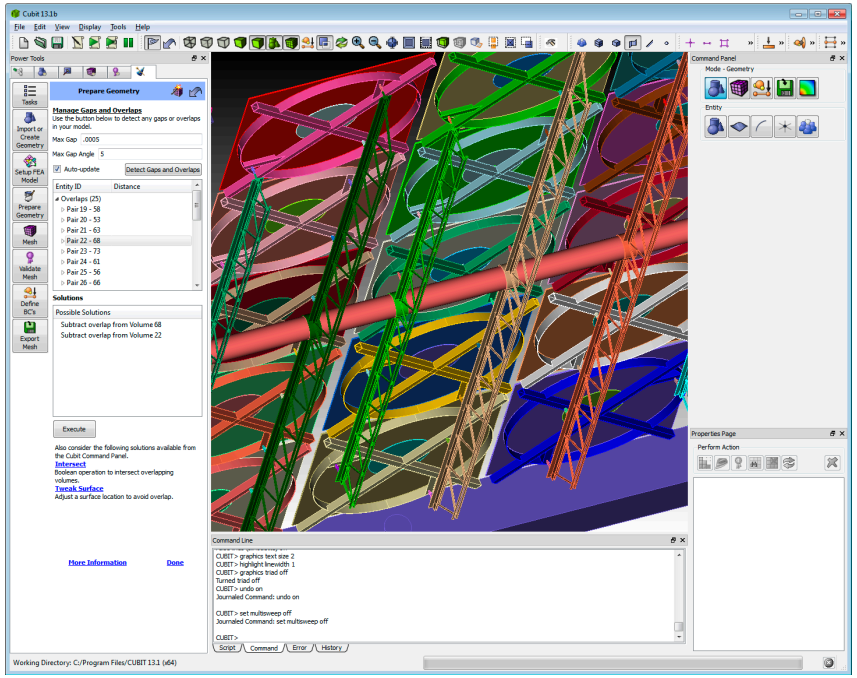


Scalability



Capacity

Complexity



COMPSIM
LIKE IT REALLY HAPPENS

Exodus

Background

- Developed 1992
- Replaced old fortran-unformatted read/write files with a
  - device-independent, random access database.
- Based on NetCDF

- Single format used by all SNL Structural and Thermal FE Codes
- Nodes, Homogenous Element Blocks, Node Sets, Side Sets
  - Transient Data on Nodes and Elements.
- A Suite of mesh generation, preprocessing, postprocessing, visualization, and translation tools developed to generate and modify exodus files. SEACAS

- Backward Compatibility Essential

Exodus Evolution

Capacity:

- 1992: CDF1 ~34 Million Nodes/Elements (~NetCDF-2.3.X)
- 2004: CDF2 ~500 Million Nodes/Elements (NetCDF-3.6.0)
  - Internal changes to exodus format to reduce dataset sizes
- 2008: NetCDF-4 (HDF5 based) 2.1 Billion Nodes/Elements
- 2012: 64-bit Integer changes (HDF5 Enabled) Multi-Billion Nodes/Elements
  - Converts integer size on the fly. Ids, indices, maps

Capability:

- Named blocks, sets, attributes, and maps
- Transient variables on all blocks and sets
- "unlimited" string size replaced fixed 32 character limit
- Full topology support (Element->Face->Edge->Node)
- Arbitrary Polyhedral element support
- Compression (via HDF5)
- File Groups (via HDF5)
- Assemblies, Entity Attributes, Discontinuous Galerkin

Exodus Evolution -- Parallel

Original workflow was file-per-processor

- External tools to split to parallel and join from parallel
- Extension library “nemesis” provided parallel data structures
- Each “processor” file is valid exodus file.

Exodus becomes “parallel-aware”

- Nemesis Embedded in Exodus (ne_? Changed to ex_?)
- PNetCDF and HDF5 provide parallel IO capabilities
- Can support auto-decomposition and auto-join

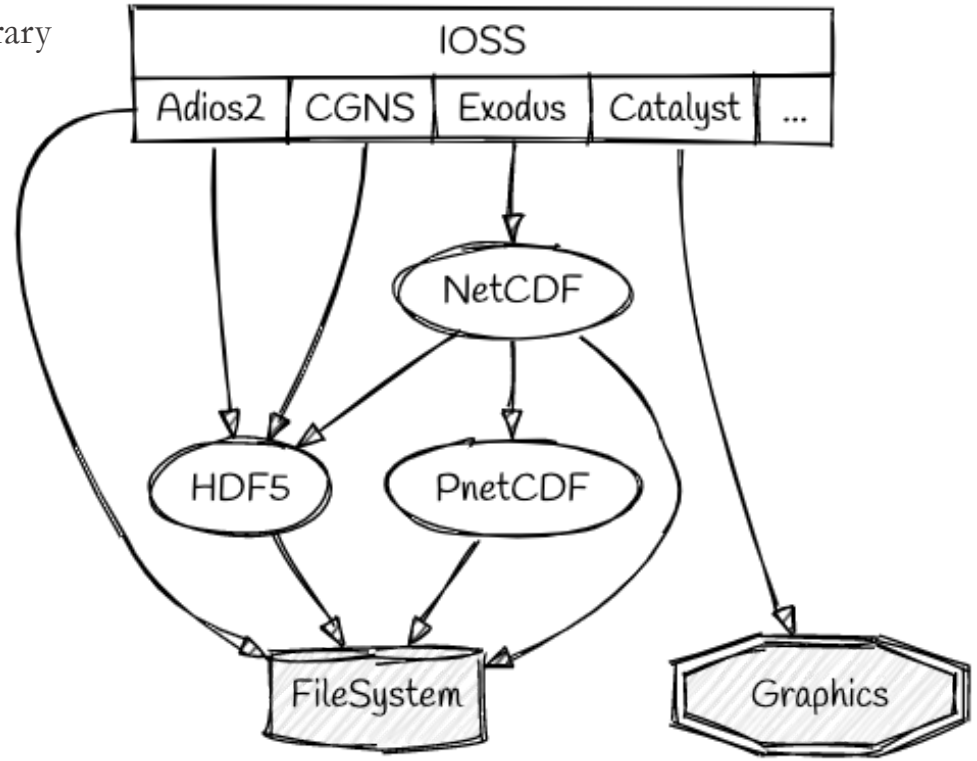
I/O Subsystem – IOSS Library

- Started as the IO component of the Sierra project – 12/1999
- Provide a database-independent interface to applications
  - (Exodus, CGNS, SAF, XDMF, ADIOS2, …)
- Also functions as a "pseudo-C++ API" for the exodus library
- Supports Advanced HPC Capabilities:
  - Kokkos Data
  - Burst Buffer
  - Data Warehouse (FAODEL)
  - Embedded Visualization

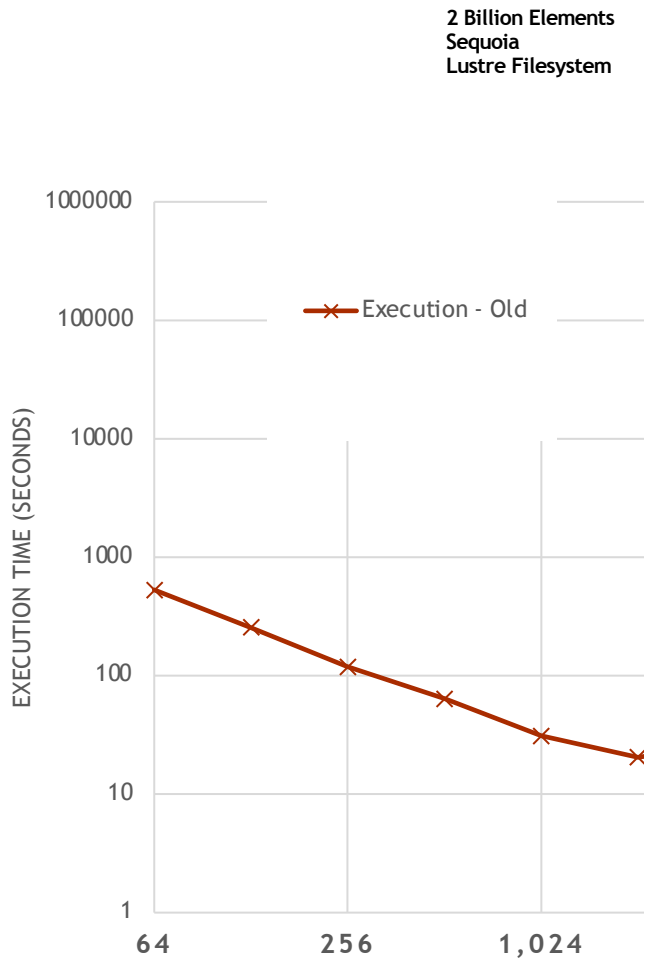
- Auto-decomposition
  - Replaces legacy file-per-processor mode
  - Uses either HDF5 or PnetCDF for parallel input
  - Uses decomposition methods in Zoltan and ParMETIS
  - Supports Exodus and CGNS (Structured and Unstructured)

- Auto-join (single file output) option
  - Uses HDF5 or PnetCDF for parallel output
  - Scalability issues.... Being addressed.

- HDF5 is used by 3 of the supported Data types.



Parallel Scalability -- Exodus

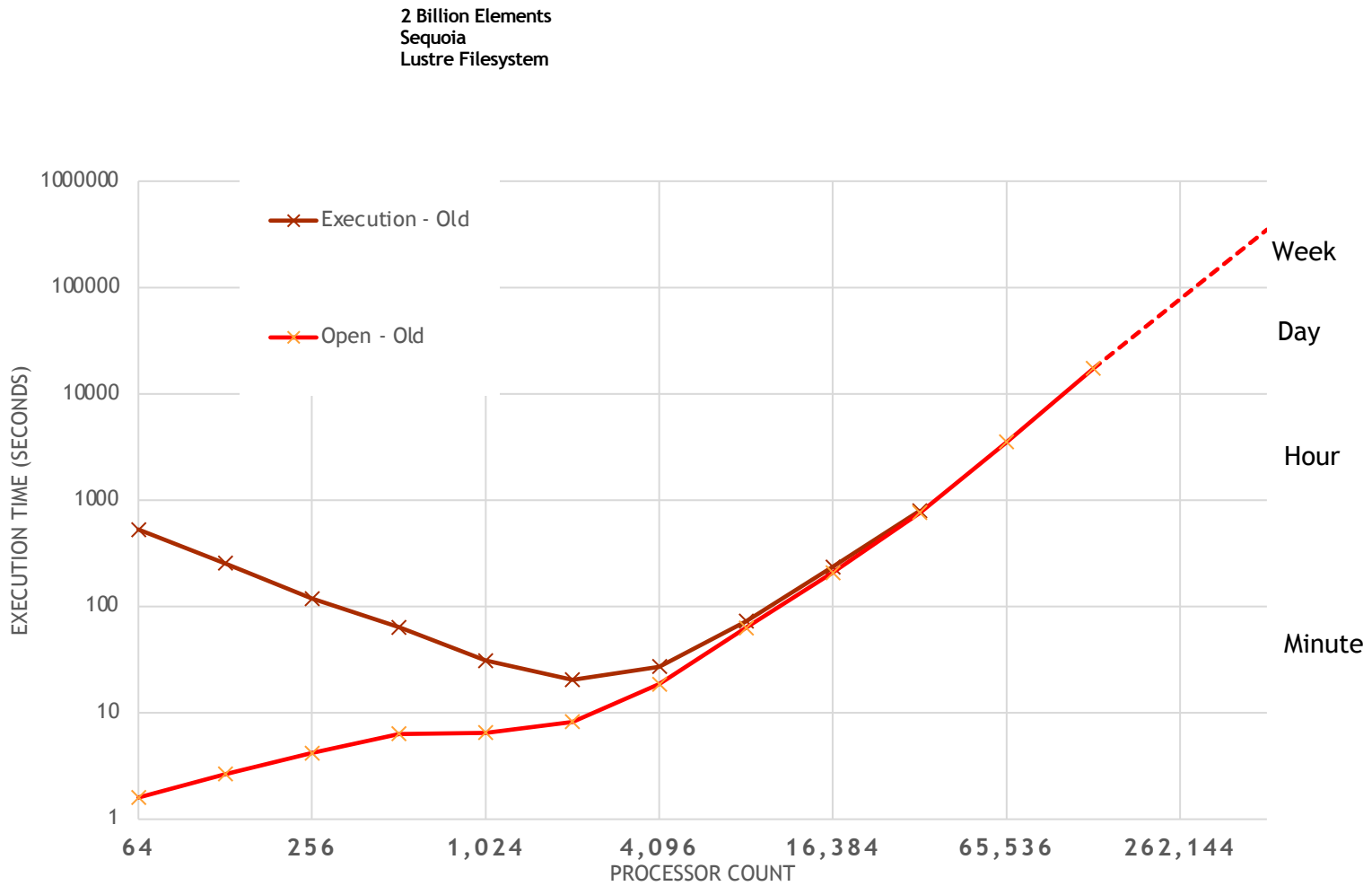


Large model being run on Sequoia
Application reported “analysis is hanging”

Ran same mesh in IOSS simulator with tracing enabled to determine what was happening

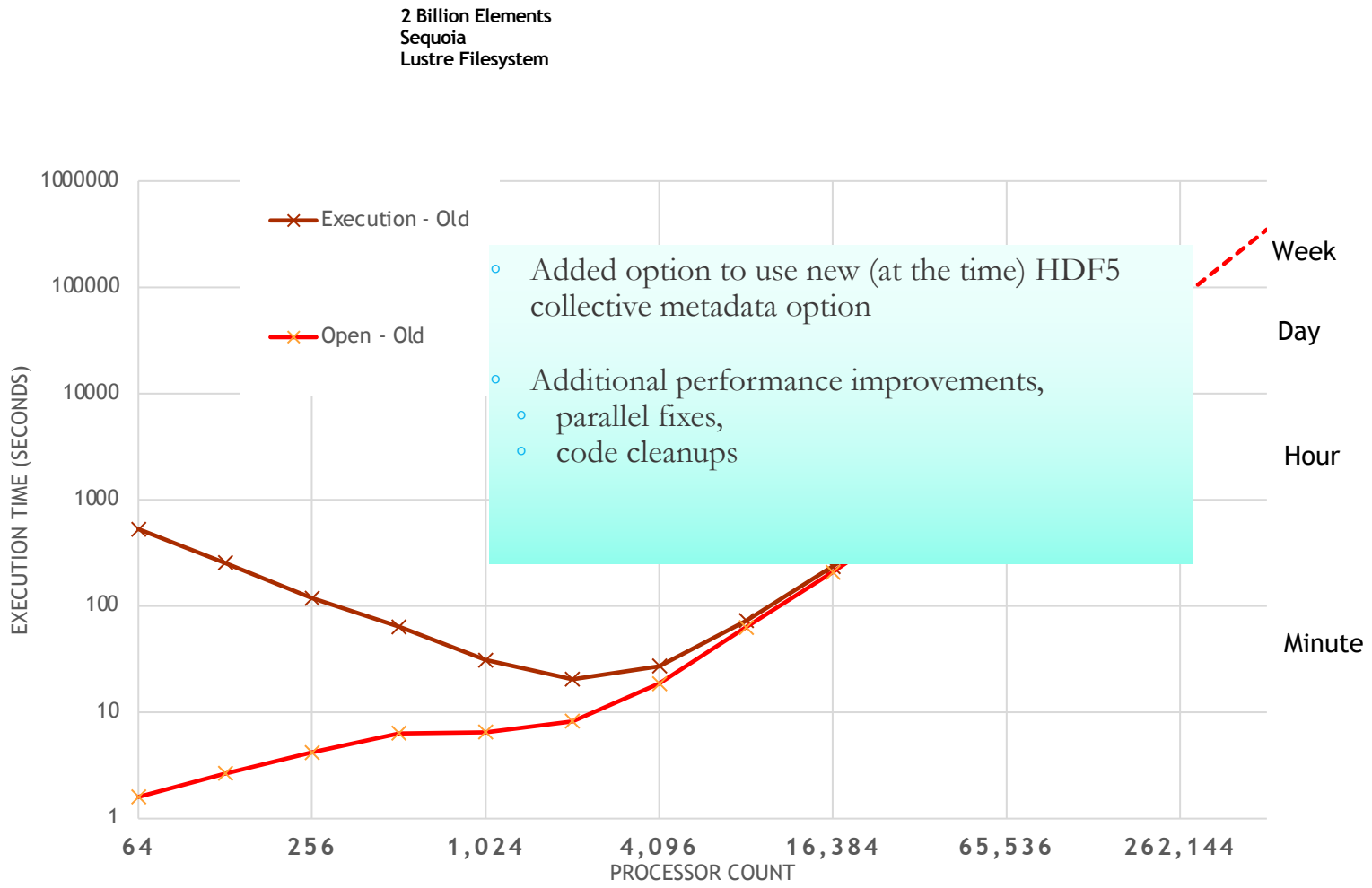
Initial results looked promising…

Parallel Scalability -- Exodus



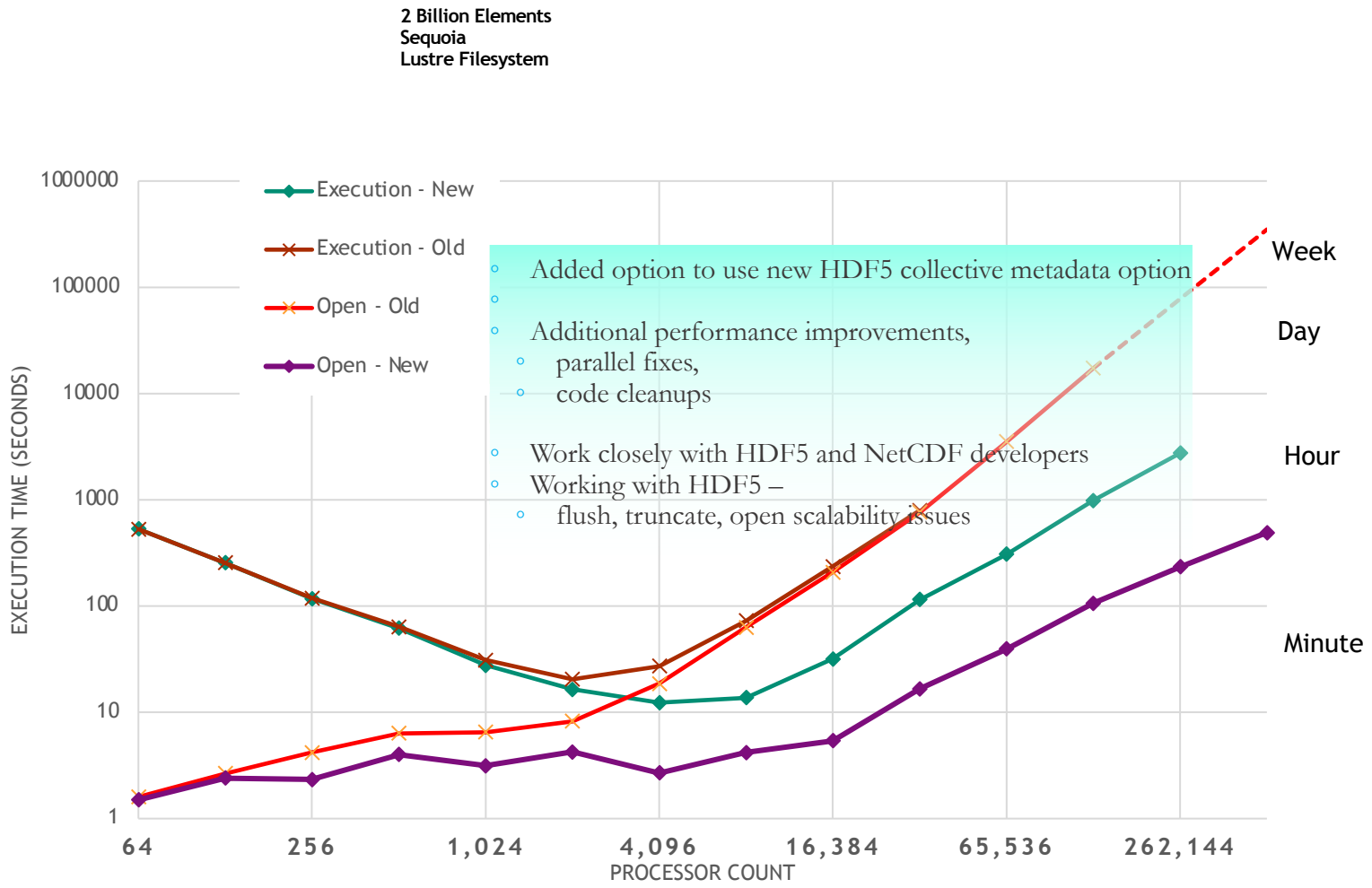
A supercomputer is a device for turning compute-bound problems into I/O-bound problems

Parallel Scalability -- Exodus



A supercomputer is a device for turning compute-bound problems into I/O-bound problems

Parallel Scalability -- Exodus



A supercomputer is a device for turning compute-bound problems into I/O-bound problems

Parallel Scalability – Exodus AutoJoin

Improving parallel scalability of auto-join (single file output)

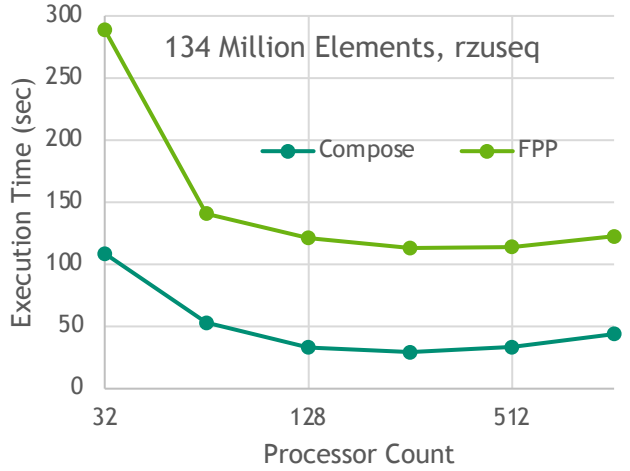
- Patch NetCDF/HDF5 to avoid unneeded data access (PR 336)
- Code review of Ioss "autojoin" routines

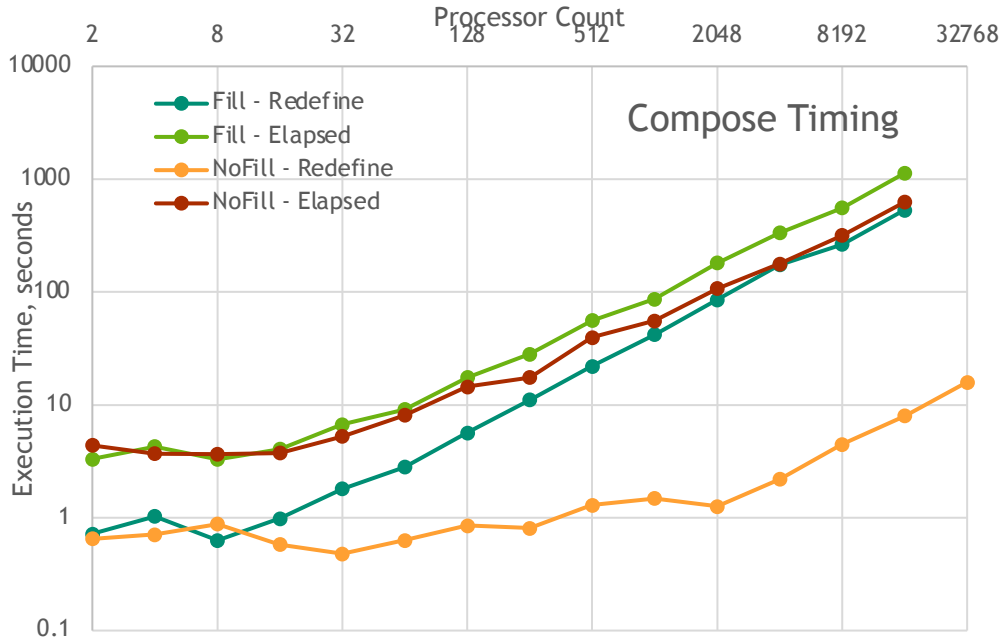
Preliminary results look promising, more to do:

- Darshan Profiling
- MPI Profiling
- MPI-IO Tuning
- Filesystem Tuning

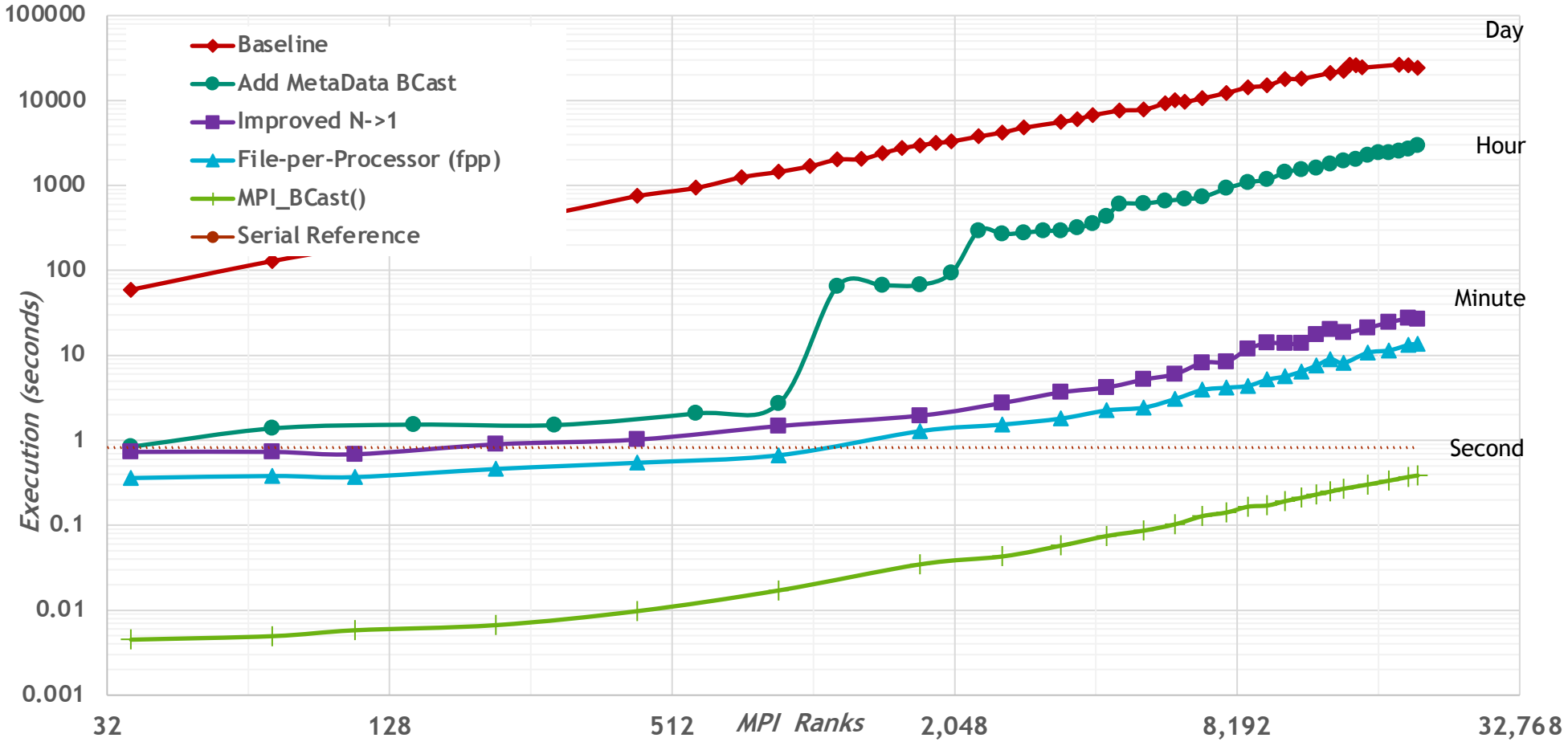
Working with HDF5 –

- Flush, truncate, close scalability issues

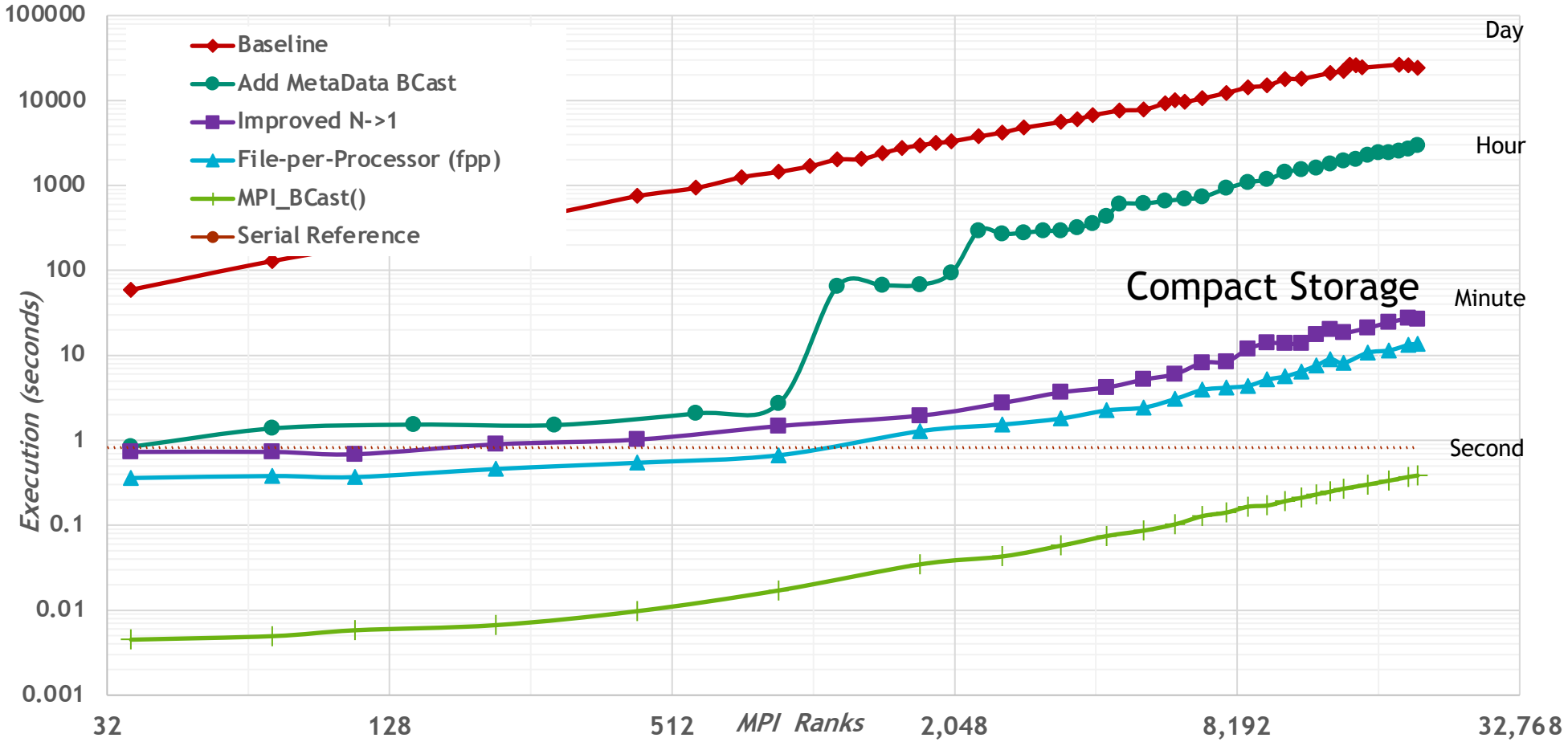




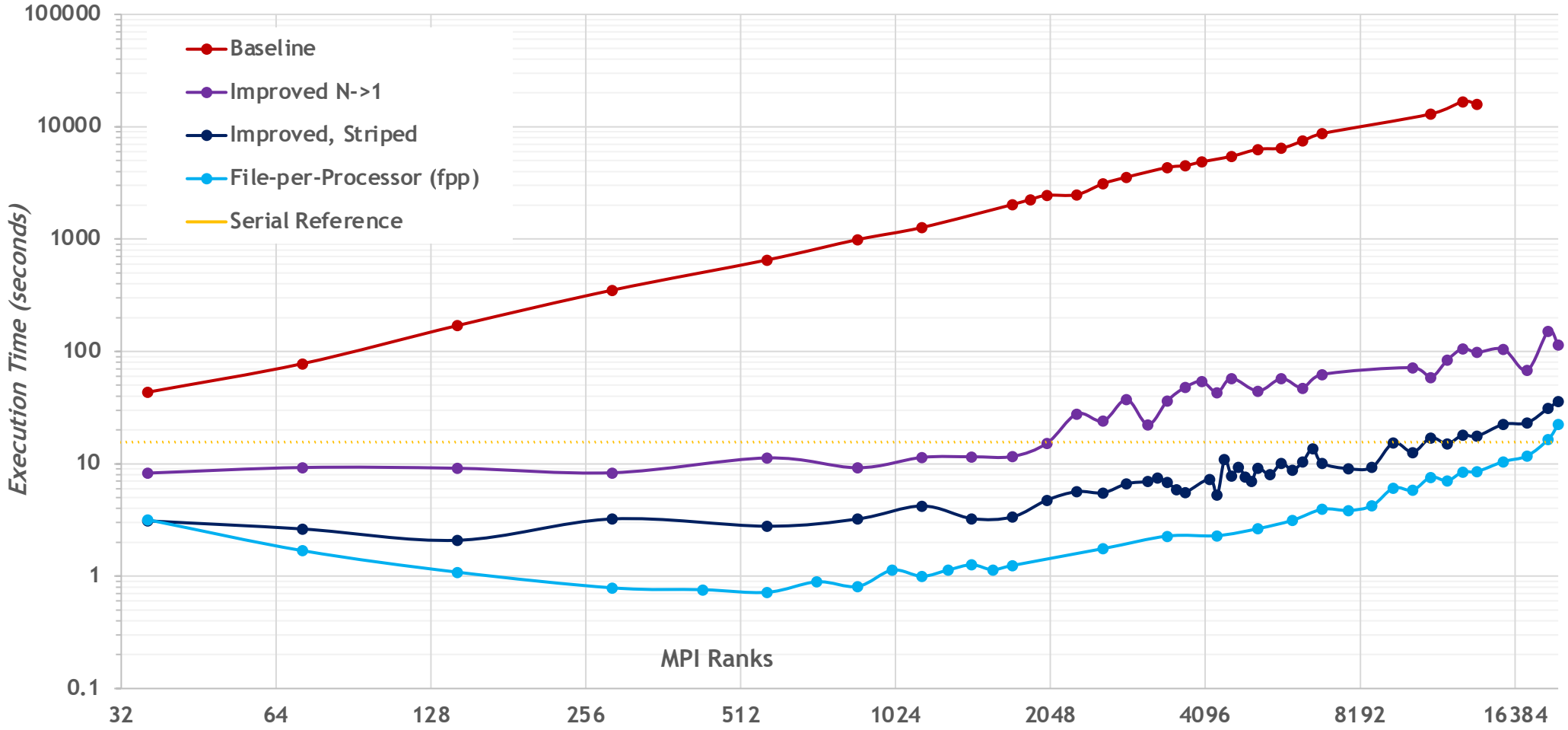
Input/Output performance improvements
CGNS Structured Mesh



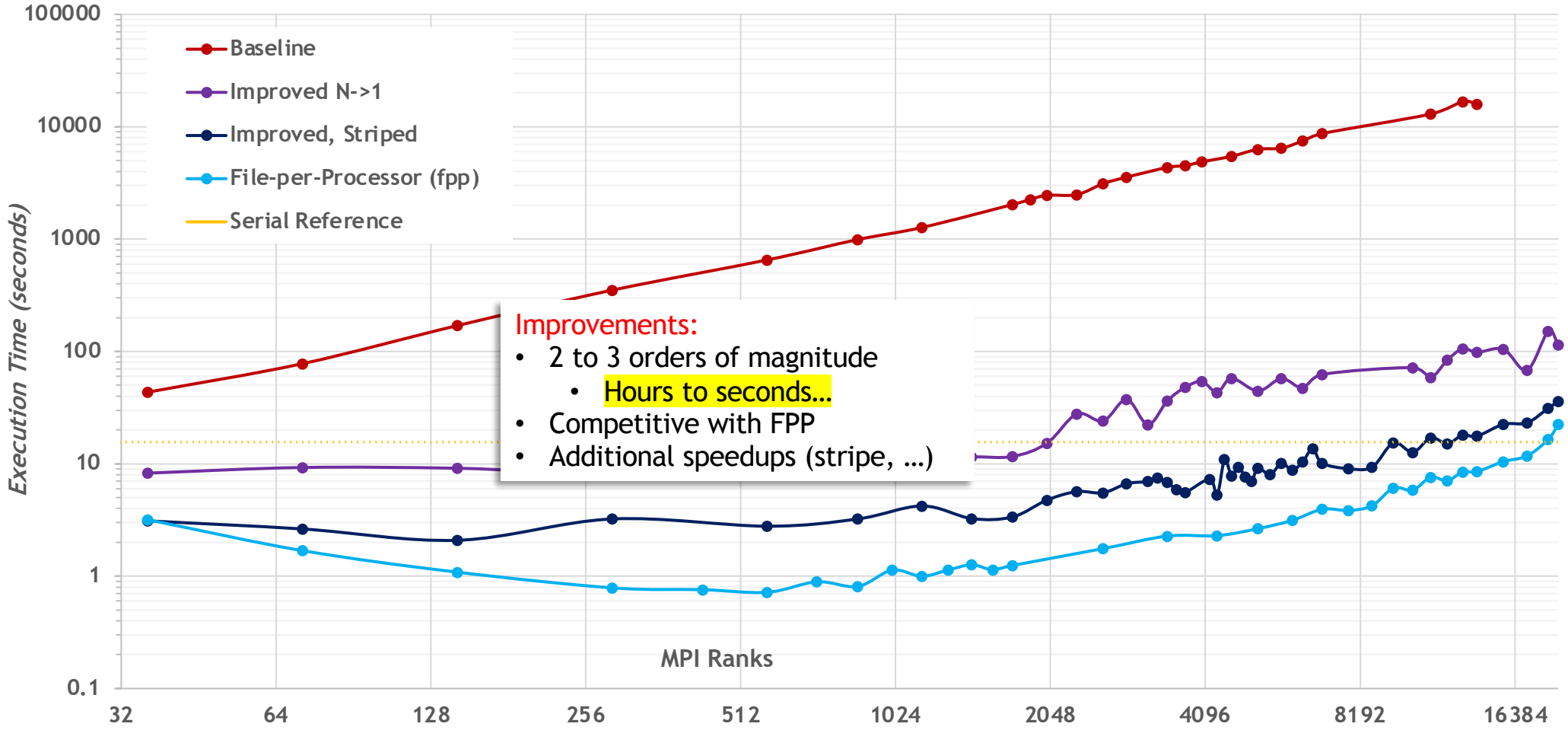
Input/Output performance improvements
CGNS Structured Mesh



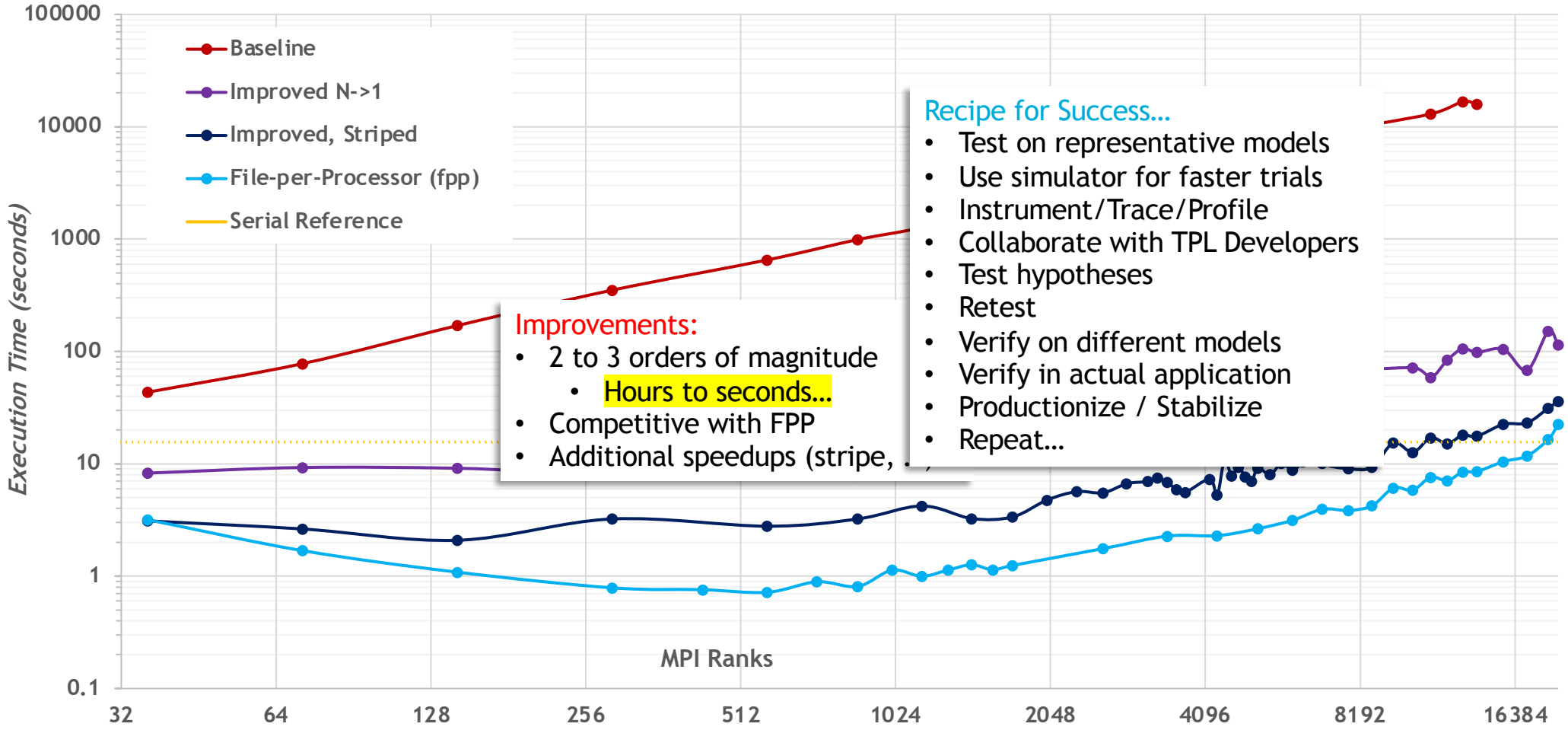
Input/Output performance improvements – Large Model CGNS Structured Mesh



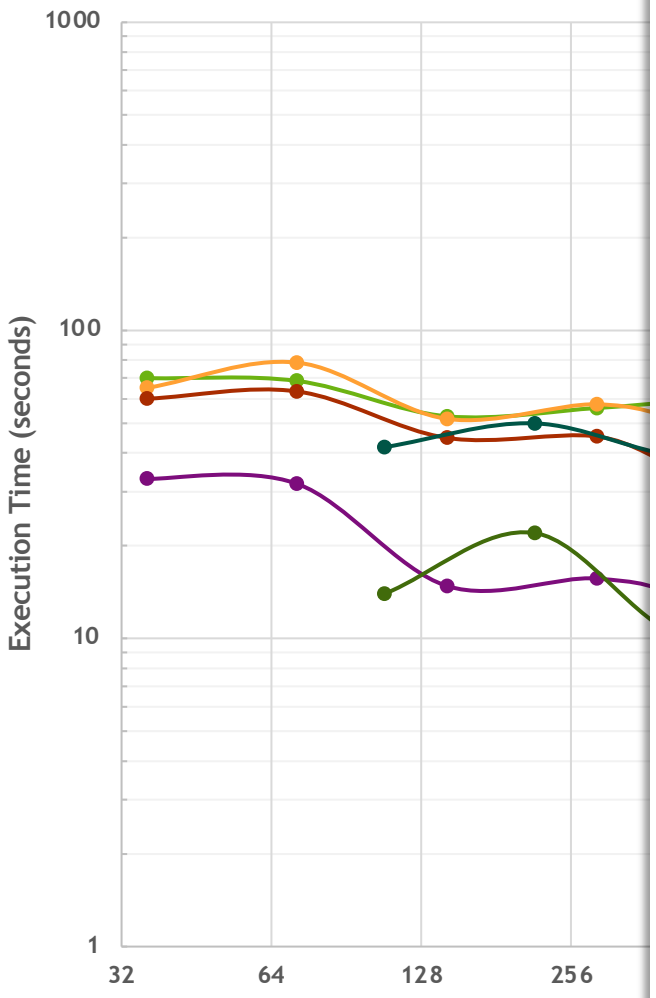
Input/Output performance improvements – Large Model CGNS Structured Mesh



Input/Output performance improvements – Large Model CGNS Structured Mesh



Exodus speedups...



Exodus format file in NetCDF-4 (HDF5) format

- 64-bit integers, 64-bit doubles
- 48.3 Gbyte file

- 390 Million finite element nodes
- 49 Million 27-node hexahedral elements
- 1 time step with 8 variables per node

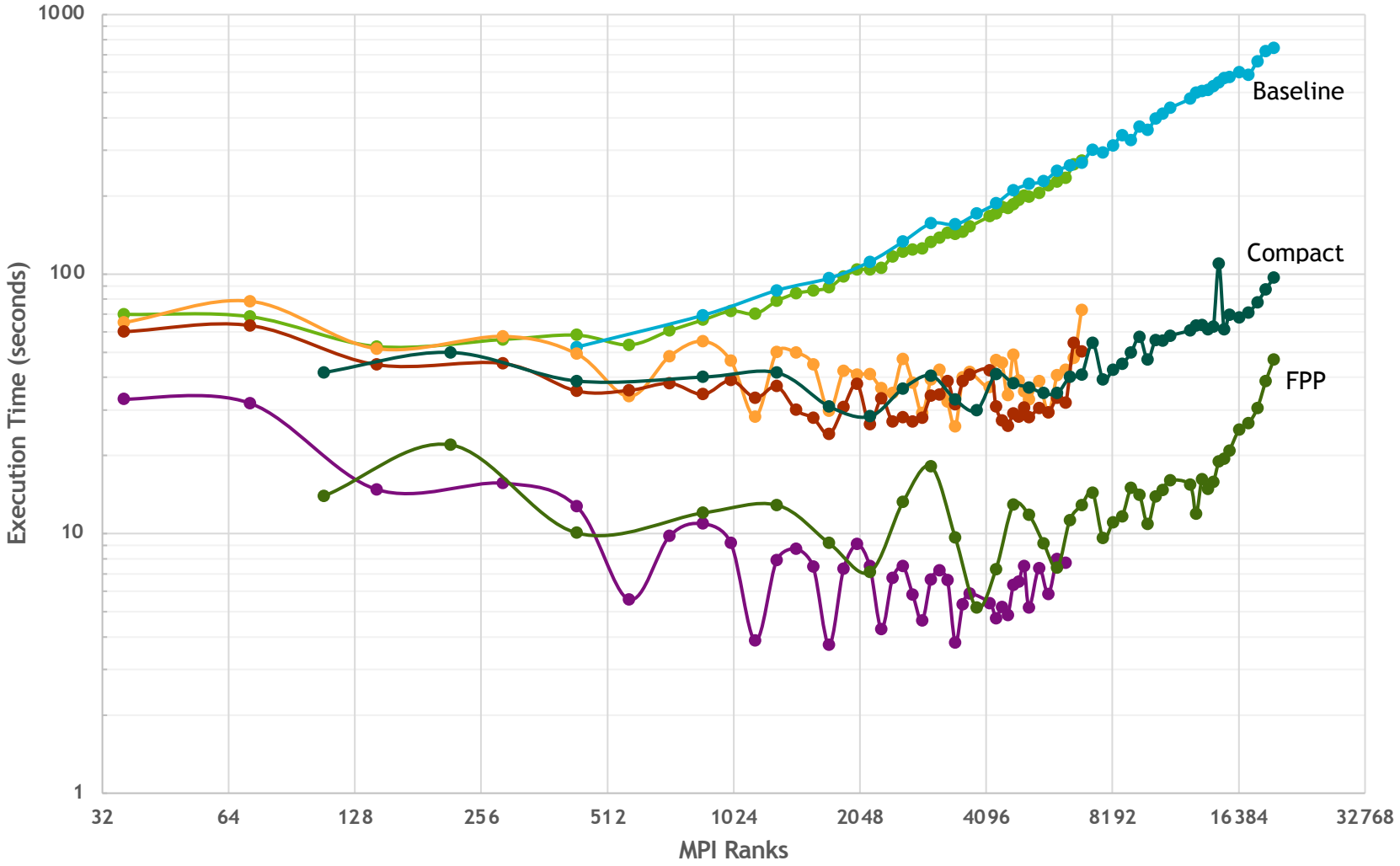
Lustre Filesystem

- “nscratch” 11 PByte
- Input file stripe count = 1
- Output file stripe count = 36

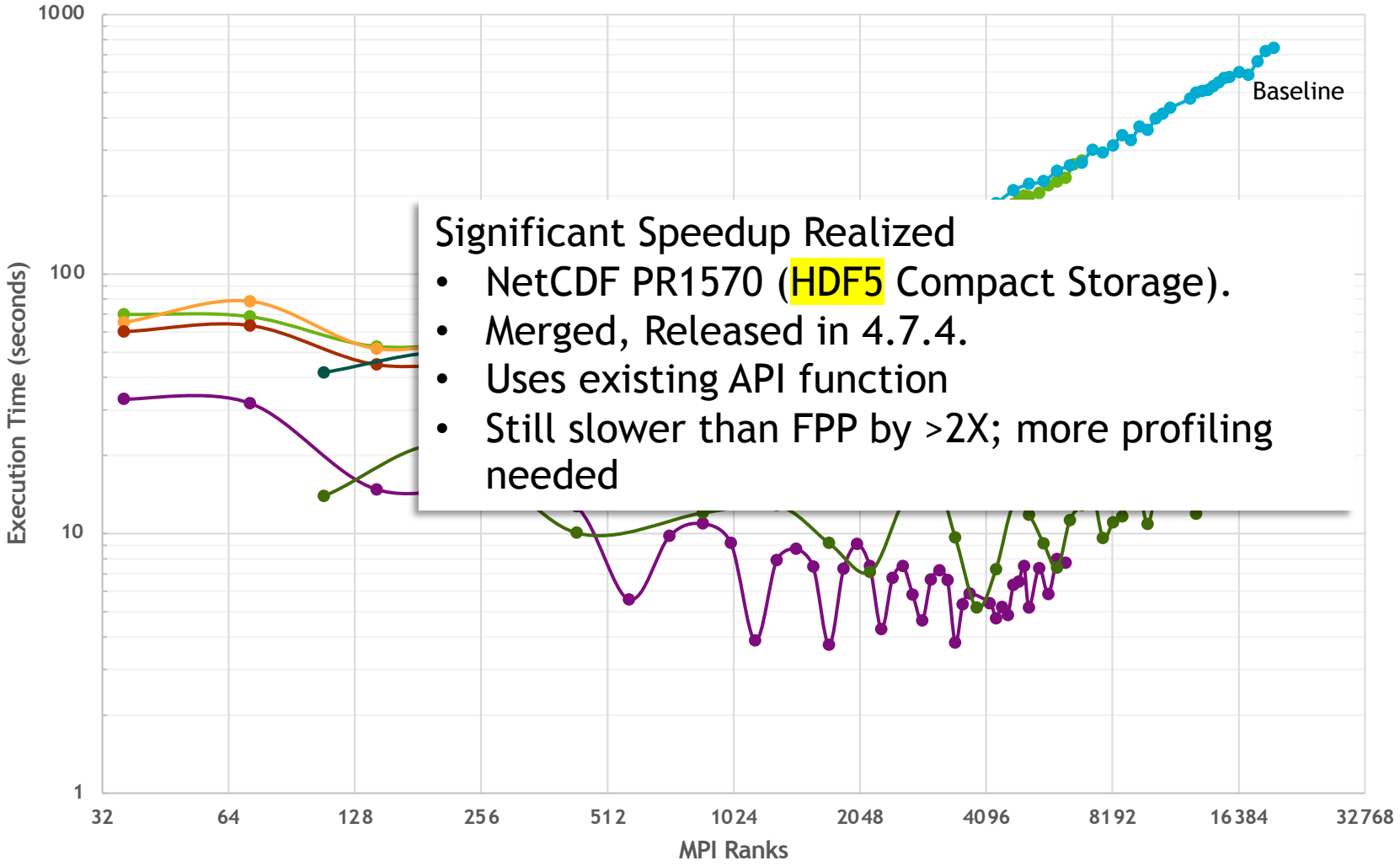
Run on “serrano” CTS-1 system

- 2.1 GHz processors
  - Dual sockets with 18 cores each
  - Intel Broadwell® E5-2695 v4
  - ~1.2 TFLOPs per node
- 128 GB RAM per node (3.55 GB per core)
- Intel Omni-Path high speed interconnect

Exodus speedups…



Exodus speedups...



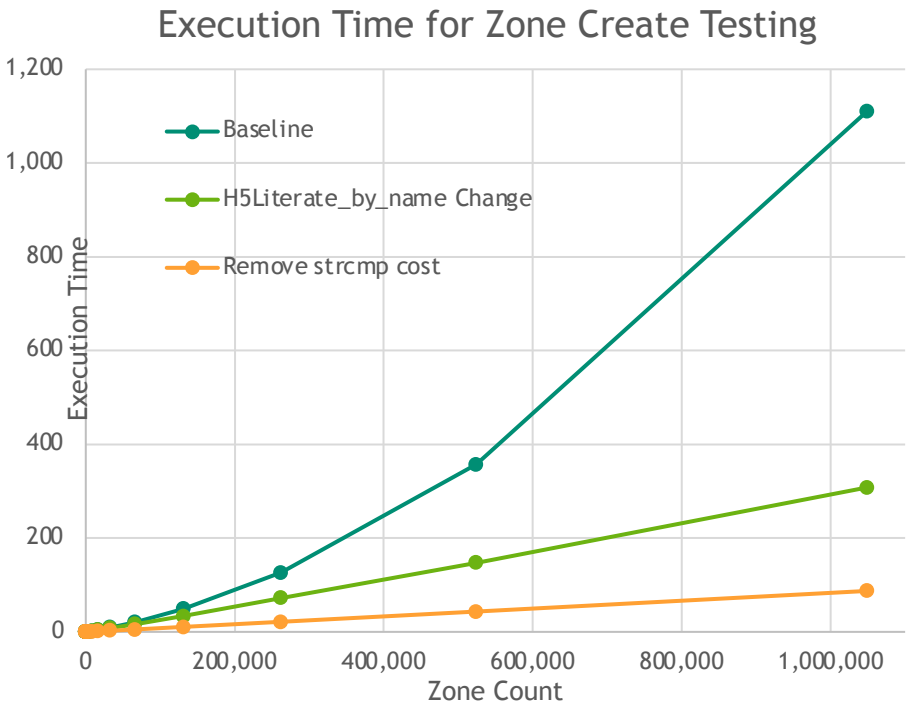
Significant Speedup Realized

- NetCDF PR1570 (HDF5 Compact Storage).
- Merged, Released in 4.7.4.
- Uses existing API function
- Still slower than FPP by >2X; more profiling needed

Cgns many zone optimization

Creating a cgns file with "many" zones exposes N^2 behavior

- Metadata only being written
- Serial or Parallel



Input by Scot Breitenfeld and Mickael Philit

Profiling showed "`H5Literate_by_name`" major contributor

`H5Literate_by_name` changed to `H5Lexists`

Code calling this function no longer shows as a hot-spot

Still N^2 behavior in another part of code due to string compares.

Looking at hashing strings to avoid this cost.

With both of these fixes, behavior is near-linear performance out to a million zones.

Conclusions

Scalability, Complexity, and Capacity are very important to HPC Applications

- Measure performance in entire spectrum; verify performance
- Make sure to probe high end of range (large MPI ranks especially…)
- Timers / Diagnostics important to locate source of non-performant behavior
- Don't guess – measure and verify.

The HDF5 library has helped Sandia I/O libraries remain performant

- **Capacity**:
  - HDF5 eliminated 32-bit limit of NetCDF, Compression
- **Complexity**:
  - HDF5 has removed several TPL limitations making it easier to store complex models
  - Use of correct function is important (H5Lexists vs H5Literate_by_name)
- **Scalability**:
  - Collective Metadata routines have reduced execution time by 2 to 3 orders of magnitude
  - Compact Storage capability has shown similar reductions
  - Burst Buffer support and other features (GPU, Threading, …) are under development and research
  - Testing of the HDF5 library at scale is being done by THG

- Thanks to Scot Breitenfeld and Elena for THG support for much of this work.