I/O Mini-apps, Compression, and I/O Libraries for Physics-based Simulations

User Productivity Enhancement, Technology Transfer, and Training (PETTT)

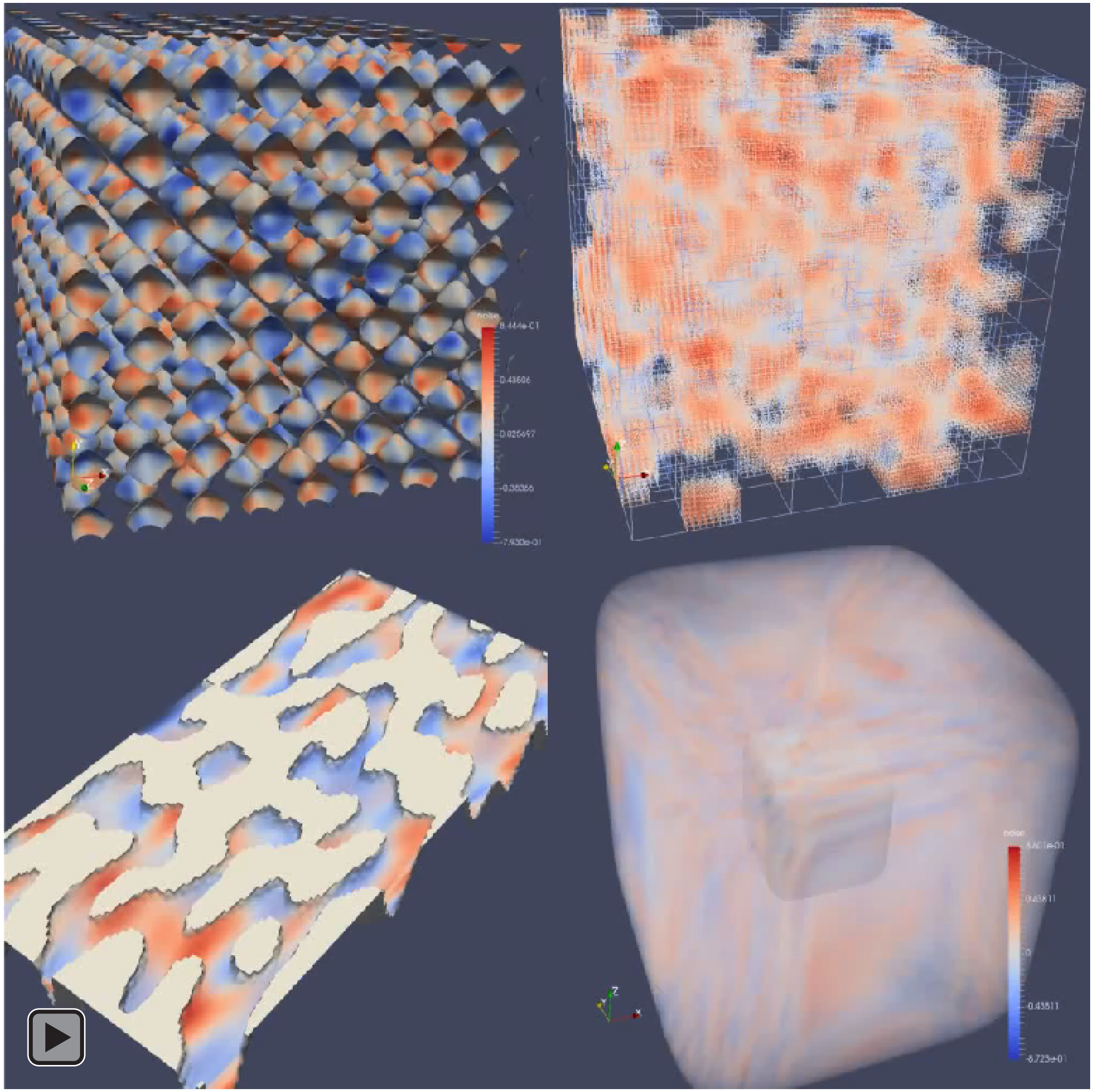
Presented by
Sean Ziegeler (Engility PETTT)
November 13, 2017

DOD HPC MODERNIZATION PROGRAM

PETTT

MiniIO: I/O Mini-apps

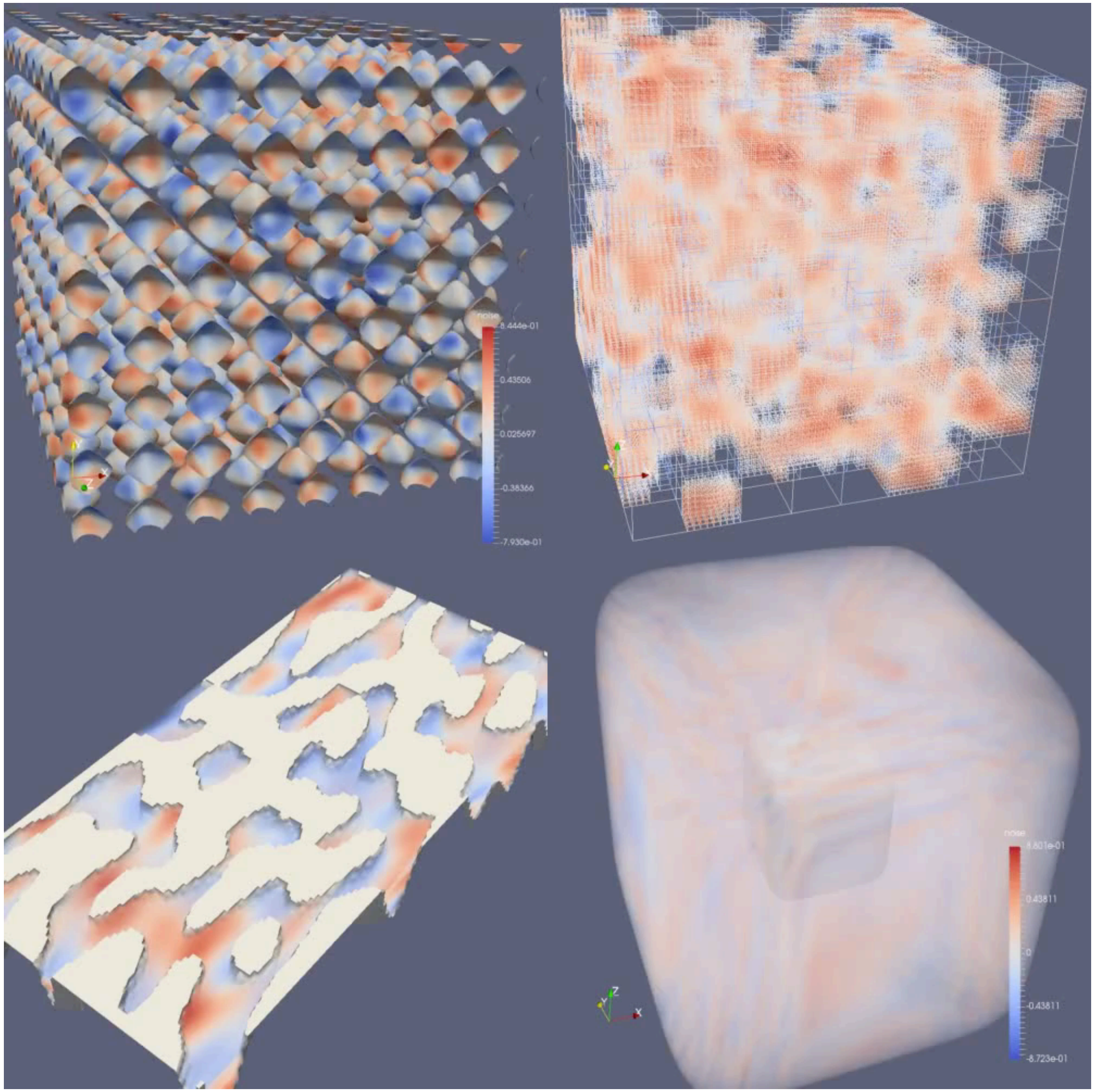
”Cartiso” ”AMR”



”Struct” ”Unstruct”

MiniIO: I/O Mini-apps

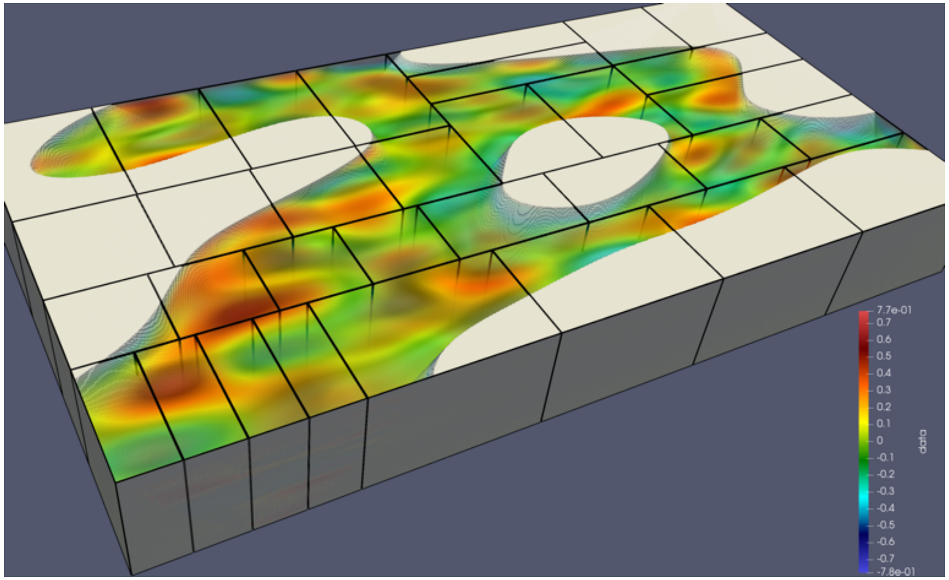
”Cartiso” ”AMR”



”Struct” ”Unstruct”

Struct Mini-app

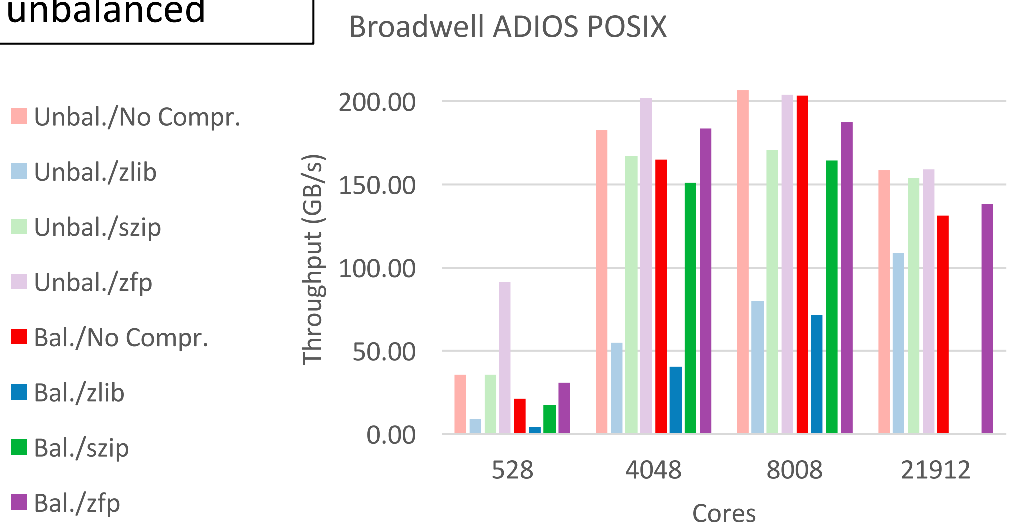
- **Struct: Structured grids with masks/blanking**
  - Masks for missing or invalid data (e.g., land in an ocean model)
    - 2D simplectic noise to generate synthetic mask maps
    - Can choose % of blanked data points
    - Noise frequency governs sizes of blanked areas (continents vs islands)
  - 4D simplectic noise to fill time-variant variables
  - Option for load balancing non-masked points evenly (as desired) across ranks
    - But creates load imbalance for I/O because blanked data is still written
    - Compression theoretically rebalances the I/O (blanked constants compress well)

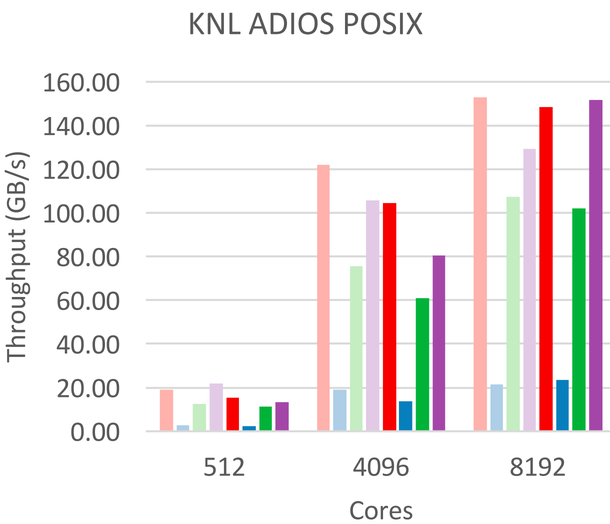


Results

ADIOS POSIX: one file per rank

Computationally unbalanced



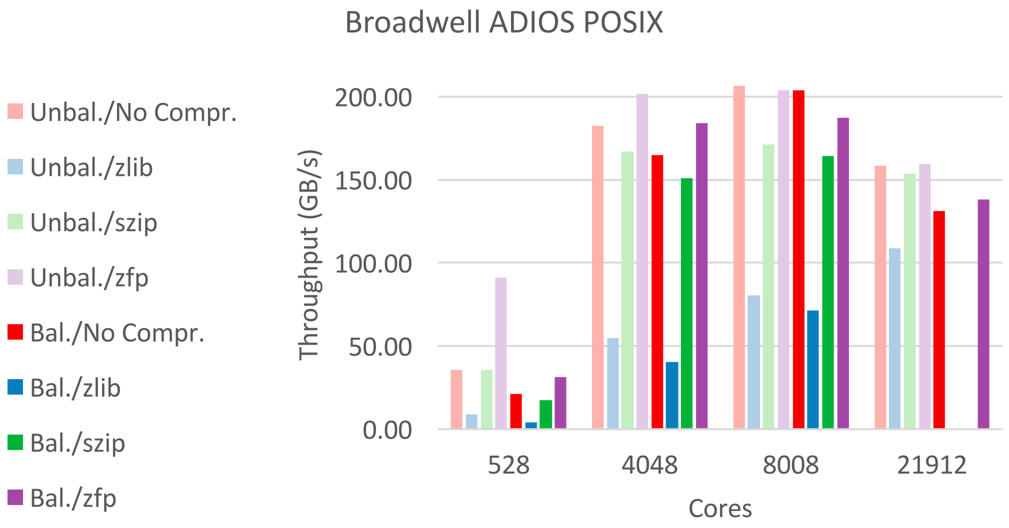


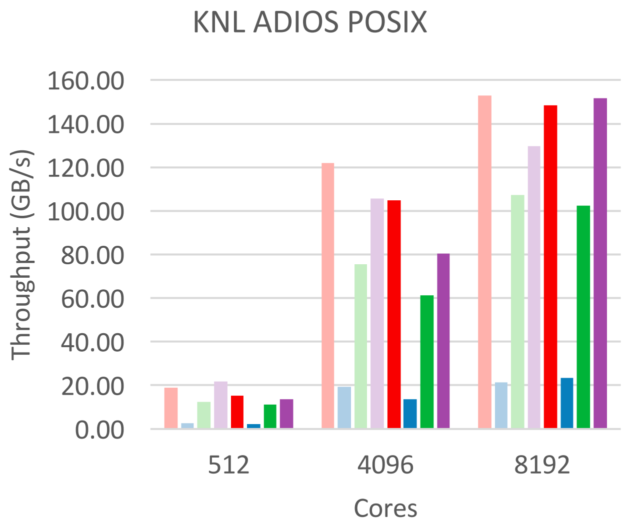
Balanced (I/O unbalanced!)

Red: No compression
Blue: zlib deflate compression (think gzip)
Green: szip compression
Purple: zfp (error bounded lossy, 0.0001), ~9:1 on average

Results

ADIOS POSIX: one file per rank

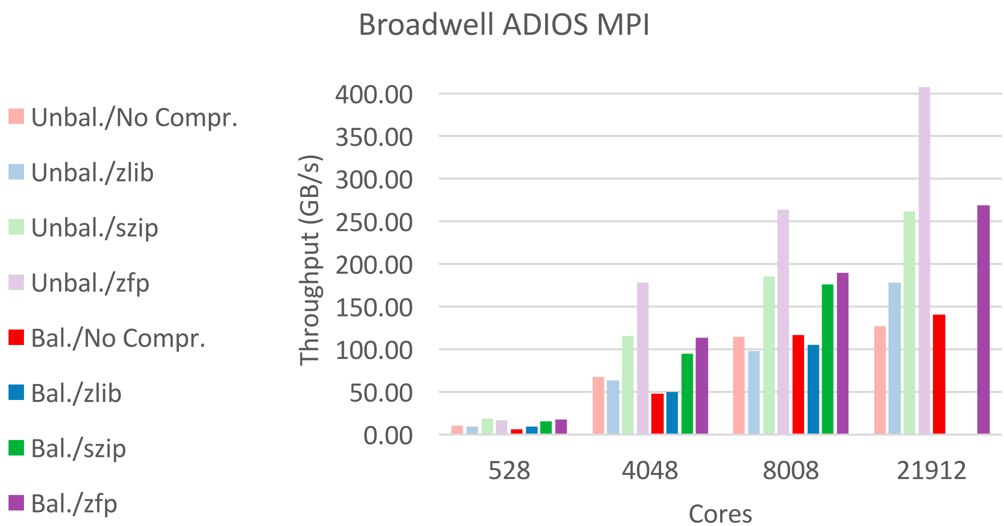


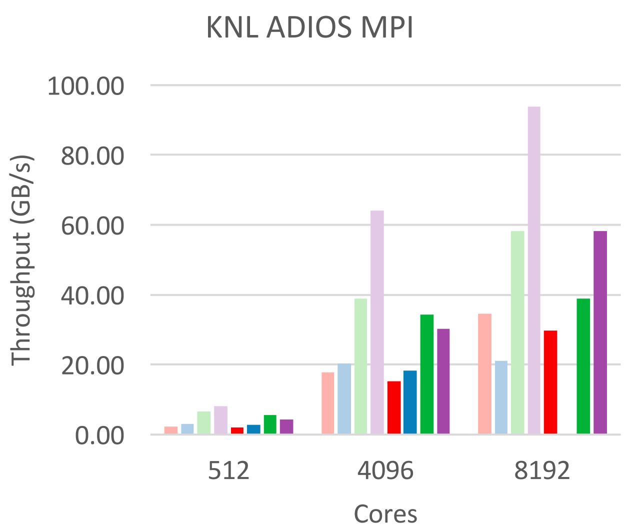


- **Initial scalability with core count**
- **Computational balancing hurts performance a little**
  - But compression sometimes helps
- **Zfp is the fastest compression**
- **KNL is slower**
- **ADIOS POSIX is the fastest without compression**

Results

ADIOS MPI: one file for all ranks





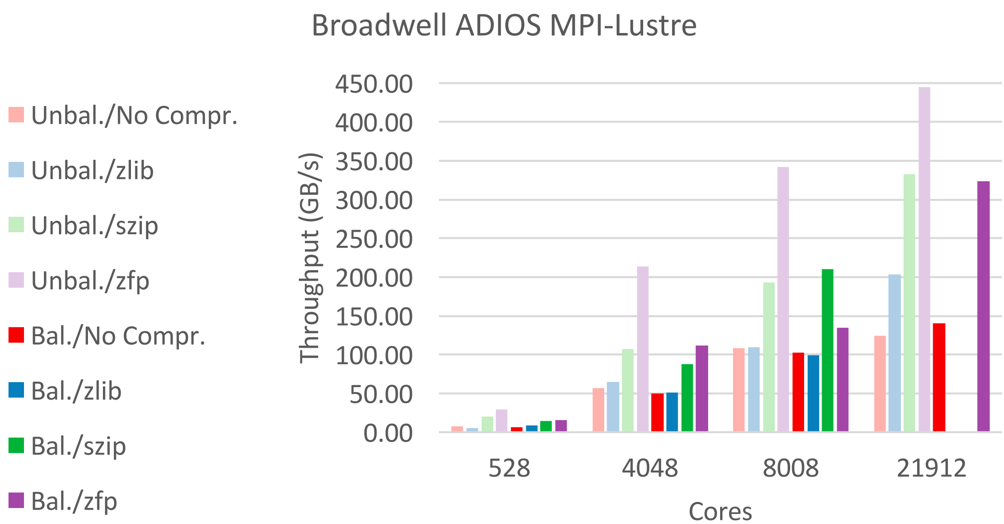
- **Good scalability with core count, especially with compression**
- **Computational balancing hurts performance a little**
  - But compression mostly helps
- **Zfp is by far the fastest compression**
- **KNL is much slower, especially the compression**

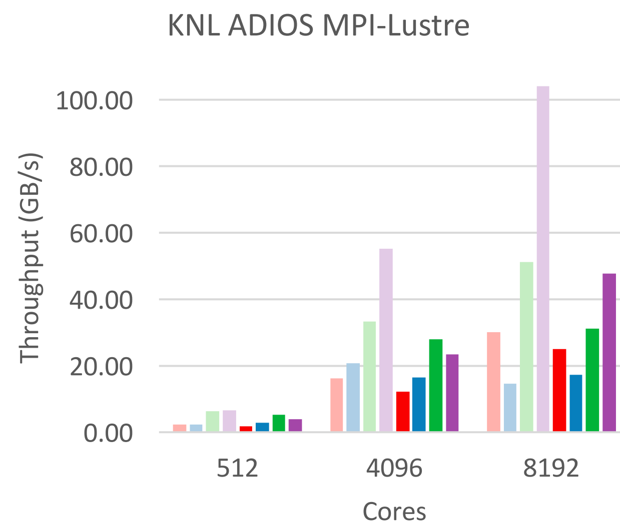
DOD HPC MODERNIZATION PROGRAM

PETTT

Results

ADIOS MPI-Lustre: one file for all ranks, tuned for Lustre file system on that system

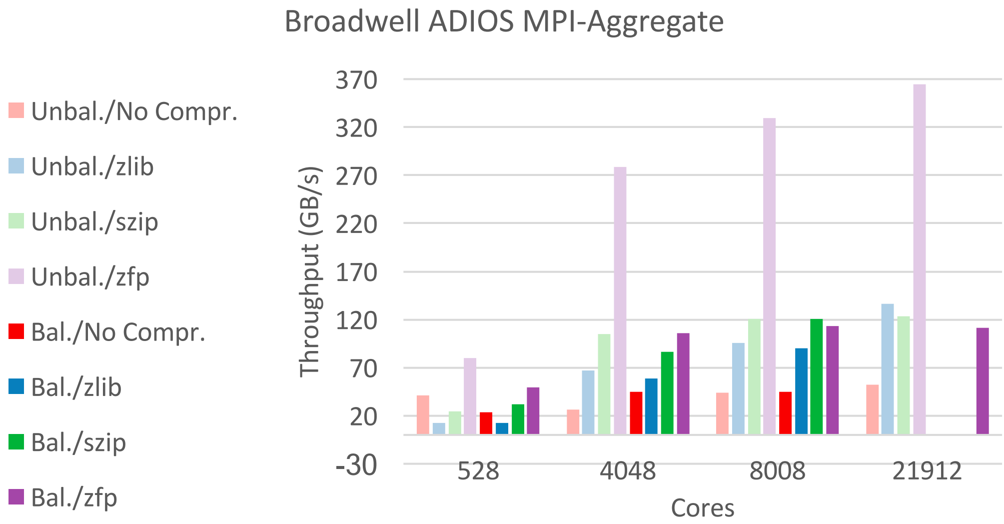


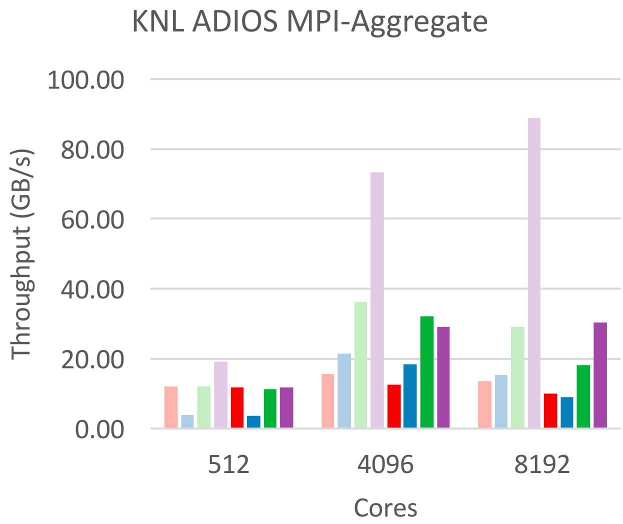


- **Good scalability with core count, especially with compression**
- **Computational balancing hurts performance a little**
  - But compression mostly helps
- **Zfp is by far the fastest compression**
- **KNL is much slower, especially the compression**
- **MPI-Lustre is the fastest with compression**

Results

ADIOS MPI-Aggregate: *m* files, *m* < number of ranks, on Lustre: *m* = #_of_OSTs

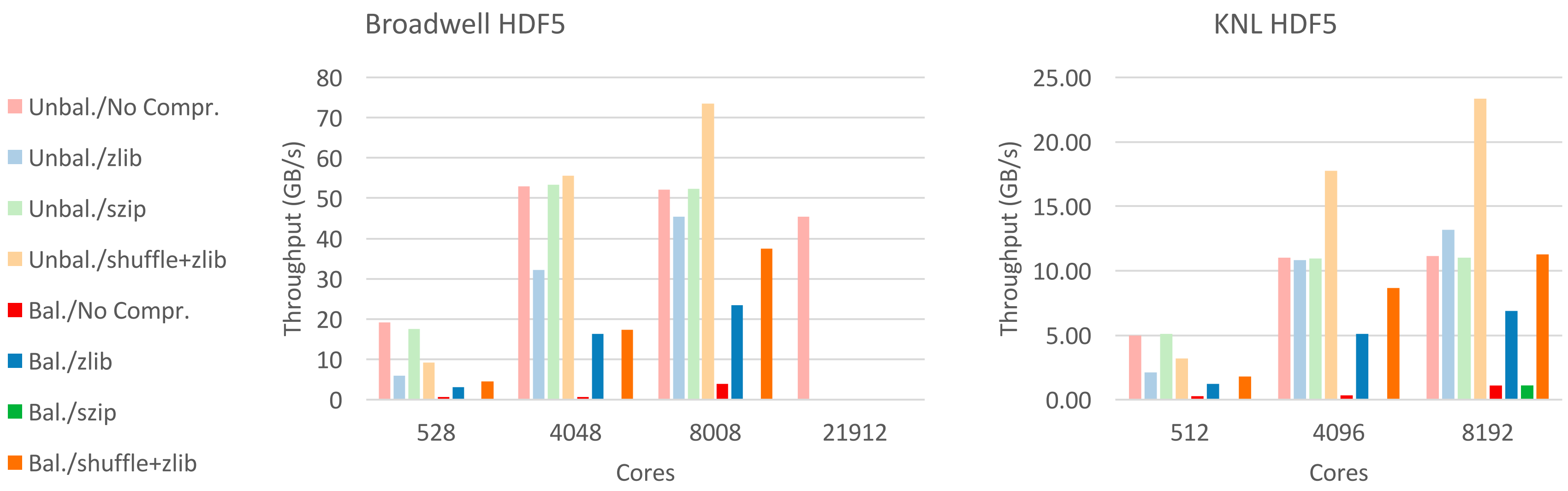




- **Good scalability with core count, especially with compression**
- **Computational balancing hurts performance very little**
  - Compression helps, but not as much
- **Zfp is by far the fastest compression**
- **KNL is much slower, especially the compression**

Results

HDF5: one file for all ranks



- **Starts slower, but scalability with core count, especially with compression**
- **Computational balancing hurts performance a lot**
  - But compression helps somewhat
- **Shuffle+zlib is the fastest compression (zfp not available at the time)**
- **KNL is much slower, especially the compression**

Conclusions

- **Compression can ”fix” I/O performance issues introduced by computational load balancing**
  - With the right output method, it is faster than unbalanced, uncompressed output
- **Compression *can* be faster than uncompressed I/O**
  - Always been theoretically possible, but rare in practice
  - Part computation: So can scale with the simulation
- **Zfp compression is very fast even at a modest compression ratio (~9:1)**
  - At scale, produces “virtual” throughput faster than the file system
  - Shuffle+zlib in HDF5 is also good
- **KNL is slower, with & without compression**
  - More cores per node ➔ fewer nodes doing parallel I/O
  - Much weaker integer processing means slower compression

Next Steps

- **Tests on Intel Broadwell cores at larger scales**
  - Complete 20k cores, begin at 40-60k cores
- **Zfp with HDF5**
- **Quilting (setting aside a few cores dedicated to I/O)**
  - Works very well for struct [separate study by SDSC] & similar apps
  - Hypothesize that quilting would be very poor for compression
  - E.g., for zfp at scale, expect that we do not want to use quilting
  - Or, at least compression on all cores, quilting after for actual I/O
- **Test on Intel Skylake cores**
  - Google Compute Engine, Gluster file system
  - 512 – 4096 cores
  - Hypothesize performance between Broadwell & KNL

This material is based upon work supported by, or in part by, the Department of Defense High Performance Computing Modernization Program (HPCMP) under User Productivity, Technology Transfer and Training (PETTT) contract number GS04T09DBC0017.

Work-in-Progress Abstract
Compiler-Assisted Scientific Workflow Optimization

Hadia Ahmed[1], Peter Pirkelbauer[2],
Purushotham Bangalore[2], Anthony Skjellum[3]

[1] Lawrence Berkeley National Laboratory
[2] University of Alabama at Birmingham
[3] University of Tennessee at Chattanooga

Introduction

Exascale Systems

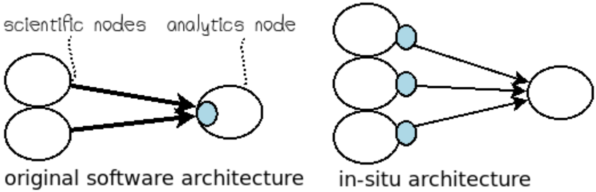
Data analytics will face tremendous challenges on Exascale systems

- Many compute nodes communicate with analytics nodes
- Simulations produce vast amount of data
- In-situ (in-transit) analytics necessary to deal with limited bandwidth
- Simulation / analytics code need to be re-organized

Idea

Describe Re-organization

Users specify re-organization with an annotation language
Tool generates optimized version

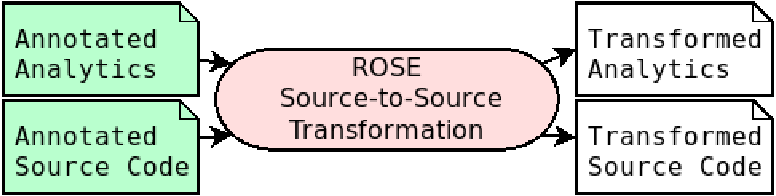


- Move code from analytics node to simulation (or vice versa)
- Describe reductions
- ...

Approach

Compiler-based

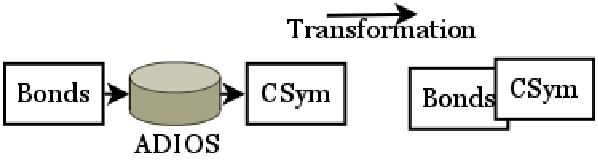
Use ROSE to read, analyze, and re-organize source files



Early Results

Restructure Bonds-CSym

On a single system, we achieved speedups between 4% and 12%.



- Restructured Bonds-CSym in a 1:1 configuration
- Re-organized code
  - Eliminates storage to file system
  - Eliminates data container conversion
  - Enables further compile-time optimizations
- Bonds-CSym is quadratic, smaller input sizes exhibit larger speedups
- Reduced need for network communication

Thank you

contact: Peter Pirkelbauer (UAB)
e-mail: pirkelbauer@uab.edu

Micro-Storage Services for Open Ethernet Drive

WIP Session

PDSW-DISCS 2017

2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems

Hariharan Devarajan, hdevarajan@hawk.iit.edu

Anthony Kougkas, akougkas@hawk.it.edu

Xian-He Sun, sun@iit.edu

ILLINOIS INSTITUTE OF TECHNOLOGY

SCALABLE COMPUTING SOFTWARE LABORATORY

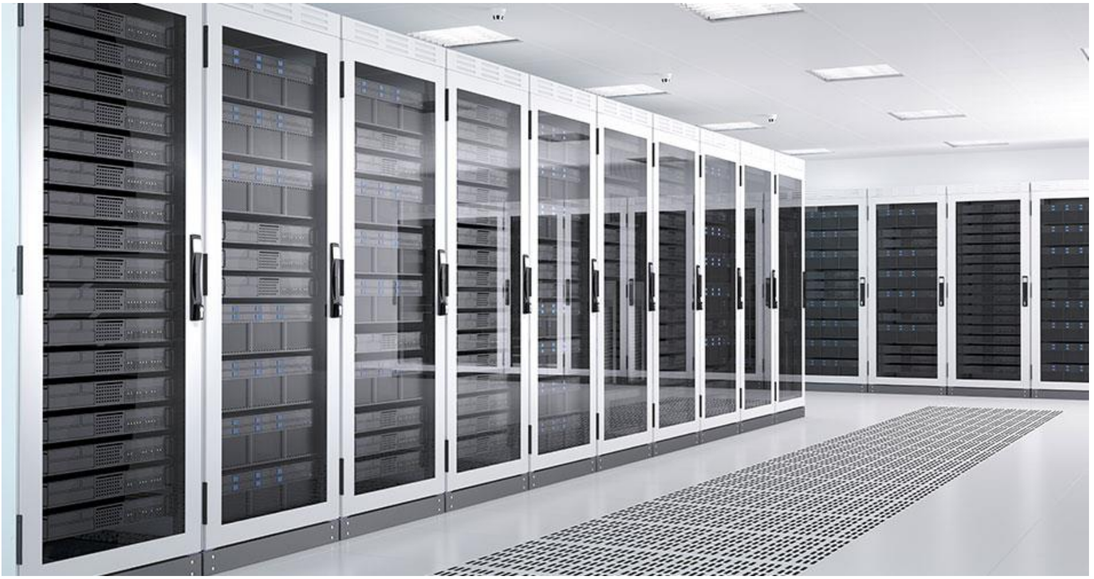
Micro-Storage Services for Open Ethernet Drive
Hariharan Devarajan, PhD Student, hdevarajan@hawk.iit.edu

Introduction

Supercomputer	K	Kaust	Tianhe-2	Trinity
# storage nodes	2000	400	1000	400

- High cost from storage
  - Purchase
  - Real-Estate (physical space)
  - Maintenance
  - Energy
    - Up to 40% of the entire energy footprint
- A very long and complex storage software stack
- Exa-scale will exacerbate this problem

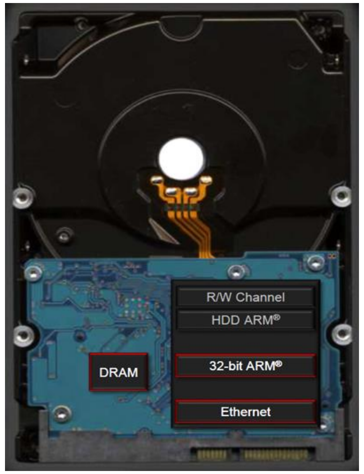


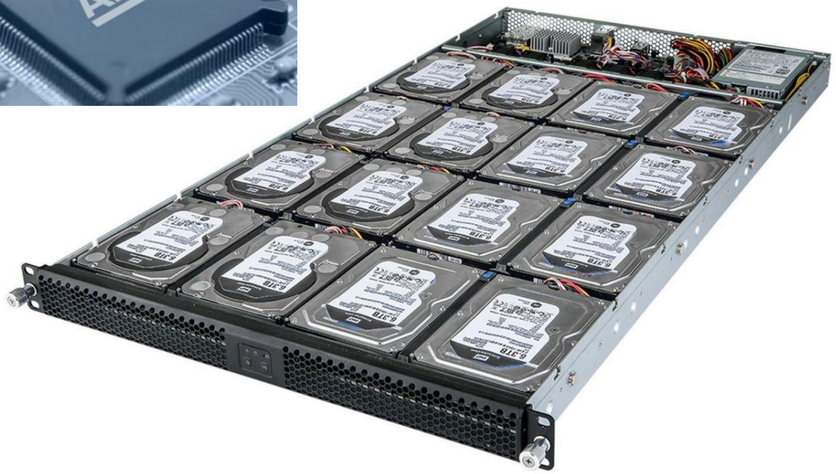


Open Ethernet Drive

- Intelligent drive
  - ARM-powered
  - Fixed sized ram
  - Network card
- Runs full-fledged Linux OS
- Prototype devices by:
  - Seagate Kinetic
  - Western Digital (HGST)
- Presented in enclosures of multiple such drives (JBOD)
- Enclosures have an embedded switched fabric (60Gbit/s)

	OED 1st Gen	OED 2nd Gen
CPU	ARM 32bit 1-core (1GHz)	ARM 32bit 2-core(2.2GHz)
RAM	2GB DDR3 1-Channel 1333Mhz	1GB DDR3 2-Channel 1600MHz
Disk	Megascale DC4000.A 4TB 7200rpm	Megascale DC4000.B 8TB 7200rpm
OS	Debian 8.0	Debian 8.1
Kernel	3.14.3	3.9
Year	2014	2016





Open Ethernet Drive - Initial results

Pros

- OEDs are capable Parallel FS and Object Store servers as well as I/O accelerators (i.e., burst buffers).
- OEDs proved to be 2.2x to 15x more energy efficient than a typical server.
- Can achieve great parallelism for the same power cap

Cons

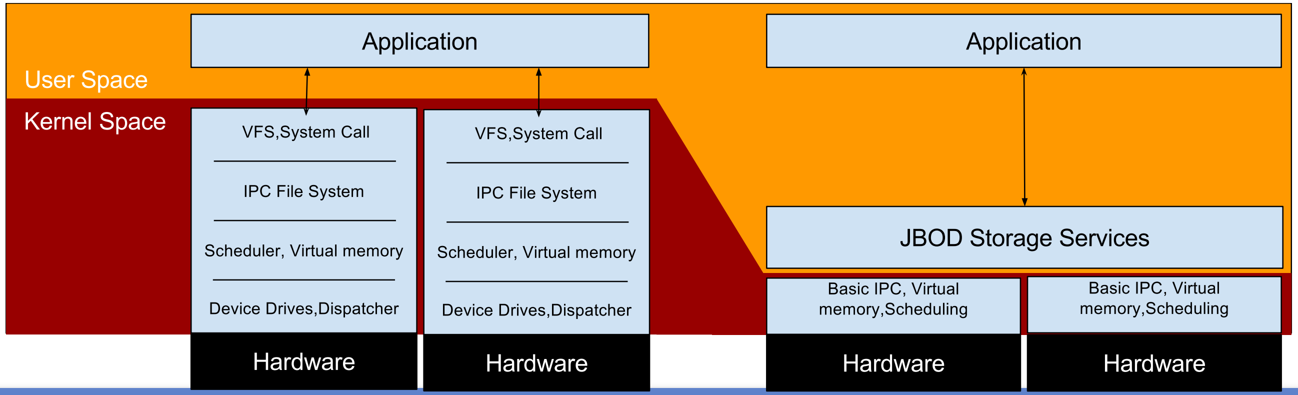
- Computation power is not at par with server nodes
- No API to use JBOD.
- Running a full-fledged Linux OS on OEDs is extremely heavy and poses unnecessary overheads

Published Work

- H. Devarajan, A. Kougkas, and X. H. Sun, "Open Ethernet Drive Evolution of Energy-Efficient Storage Technology." in Proceedings of DataCloud'17, Denver,CO.
- A. Kougkas, A. Fleck, and X. H. Sun, "Towards energy efficient data management in HPC: The open Ethernet drive approach," in Proceedings of PDSW-DISCS'16: 2017, pp. 43–48.

Proposal – Design Objectives

- Micro storage kernel
  - Minimize OS unnecessary overheads.
    - Modules which are not crucial to storage nodes would be removed.
  - Maximize performance
    - Fine-tune the kernel to better suit the needs of the OED technology
- Lightweight API
  - Maximize utilization of JBOD
  - Parallelization of I/O tasks
  - Offload small computation to JBOD
  - JBOD Services:
    - Manager, I/O Scheduler, Load Balancer
  - Provide mount point for application



Our first steps

- BusyBox 1.27.2 Linux
  - As a building block
  - Very small size (i.e., ~5MB)
  - Add XFS file system
- Results
  - Reduced boot time by 1300%
  - Smaller memory footprint leading to more available memory to applications (i.e., from 350MB to only 15MB)
- Next step:
  - Investigate other lightweight Linux distributions for embedded and mobile platforms (e.g., ToyBox)
  - Develop a light-weight parallel file system within the JBOD.

Micro-Storage Services for Open Ethernet Drive

Hariharan Devarajan, hdevarajan@hawk.iit.edu

Comprehensive Burst Buffer Evaluation

Eugen Betke, Julian Kunkel

Research Group
German Climate Computing Center
2017-11-12

Objectives

- Understanding how burst buffers can be used in an alternative way
  - Burst buffers are mainly used for catching I/O peaks
- Improving runtime of I/O intensive application by better workflows
- Reducing procurement costs by intelligent usage of burst buffers

Test systems and evaluation tools

Test systems

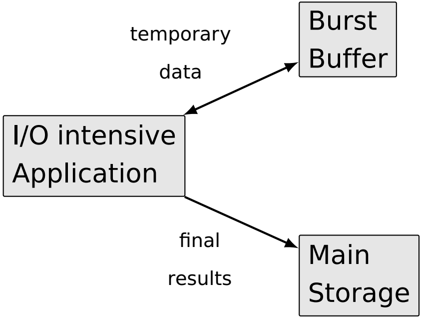
- Kove XPD [3]
  - In-memory storage
- DDN IME [5]
  - SSD-based
- Cray DataWarp [2]
  - SSD-based

Parallel I/O benchmark tools

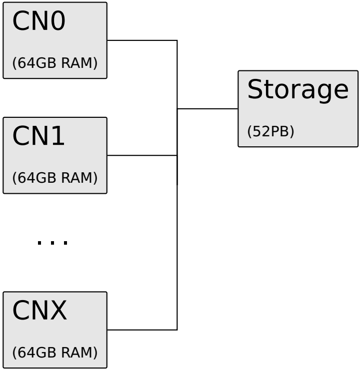
- NetCDF-Bench [4]
  - is a parallel NetCDF benchmark
  - generates I/O load to a shared NetCDF file
  - mimics scientific data
    - Many climate scientist favor NetCDF to other formats, because it offers powerful features and has a simple interface.
- IOR
  - uses MPI-IO interface in our tests
  - generates I/O load to individual files in order to get best I/O performance

Short-term campaign storage space

- Purpose
  - Reduction of I/O load on main storage
- Basic idea
  - Storing temporary data on main storage may be inefficient when
    - Temporary data is stored on burst buffer
    - Results are stored on main storage
- Expectation
  - Speed up of I/O intensive applications
- Evaluation methodology
  - Gathering of burst buffer characteristics
- Goal
  - Intelligent and efficient workflows



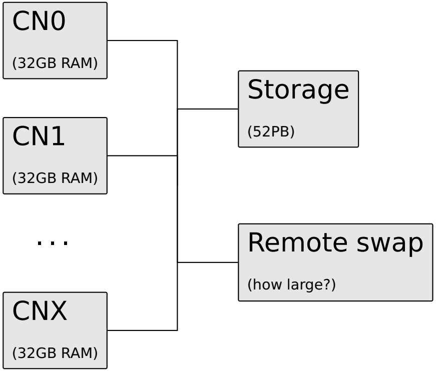
Reducing procurement costs of HPCs [1]



Observations made on Mistral [1] (HPC of DKRZ)

- Most applications are using only a fraction of available memory
- A few memory intensive applications have high memory requirements

Reducing procurement costs of HPCs [2]



- Purpose
  - Reducing total HPC costs
- Basic idea
  - Equip compute nodes with less memory
  - For memory intensive application use remote swap file system
- Expectation
  - Most programs are not affected
  - Memory intensive application are affected by swap overhead
- Evaluation methodology
  - Tracing of swap in/out with kprobes
- Goal
  - Cost model

References

- **HLRE-3 "Mistral"**. https://www.dkrz.de/Klimarechner/hpc. Accessed on 2017-03-22.
- Cray Inc. **Cray XC40 DataWarp's applications I/O accelerator**. Cray Inc. Cray Inc. 901 Fifth Avenue, Suite 1000 Seattle, WA 98164, Oct. 2015.
- Kove. **Kove XPD**. http://kove.net/downloads/Kove-XPD-L3-datasheet.pdf. Accessed on 2017-08-24. 2017.
- **NetCDF-Bench**. https://github.com/joobog/netcdf-bench. Accessed on 2017-08-25.
- DDN Storage. **Burst buffer & beyond; I/O & Application Acceleration Technology**. DDN Storage. Sept. 2015.

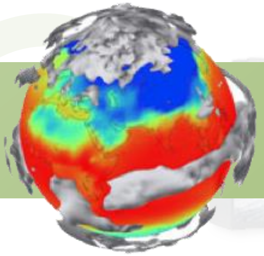
S. Di Girolamo, P. Schmid, T. Schulthess, T. Hoefler

SimFS: A Simulation Data Virtualizing File System

Systems@ETH zürich

Disk-Backed Solution

MegaBytes
1995



PetaBytes
2015

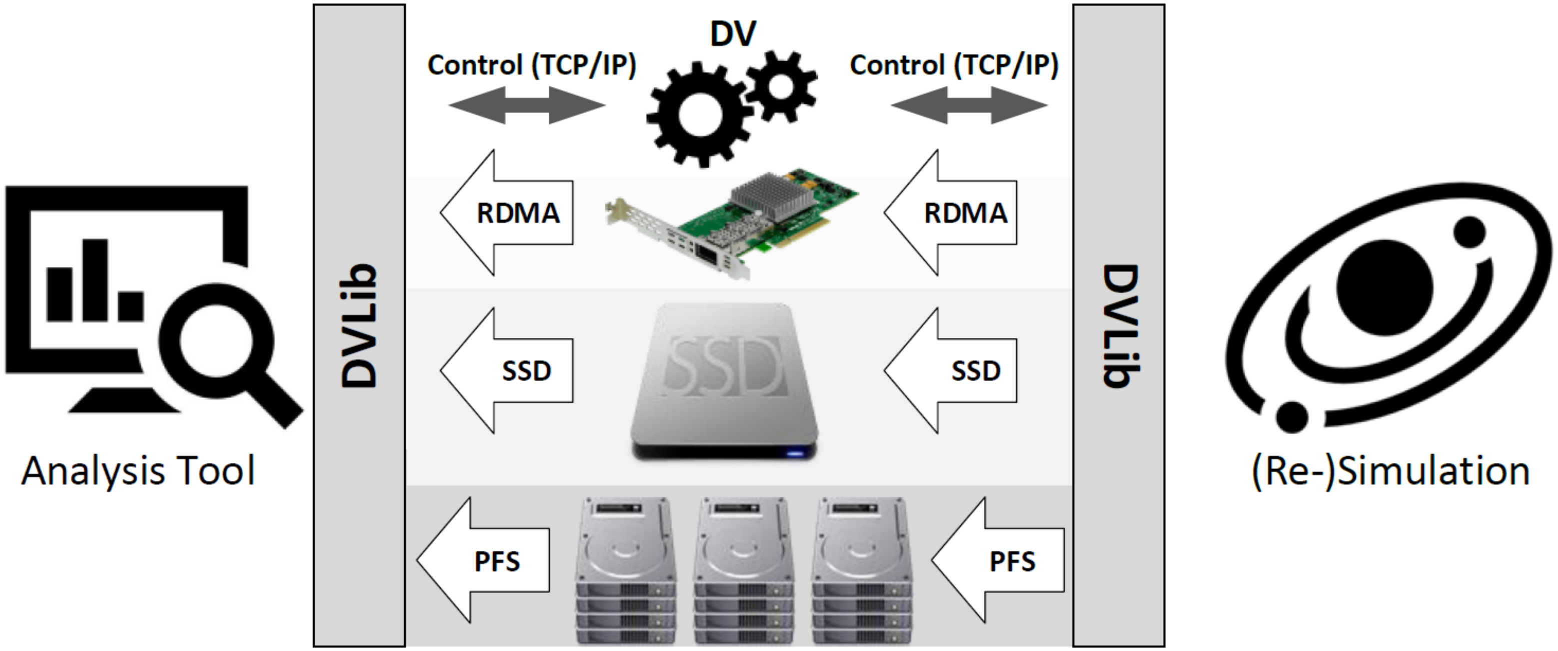
ExaBytes?
2025

Elasticity
Persistent data
I/O Capacity

Maintenance Cost: 100$/TB/year

Exabyte/year cost: 100'000'000$

SimFS: Virtualizing Simulation Data



Backup

In Situ Solution

1 Runs the simulation

Simulator

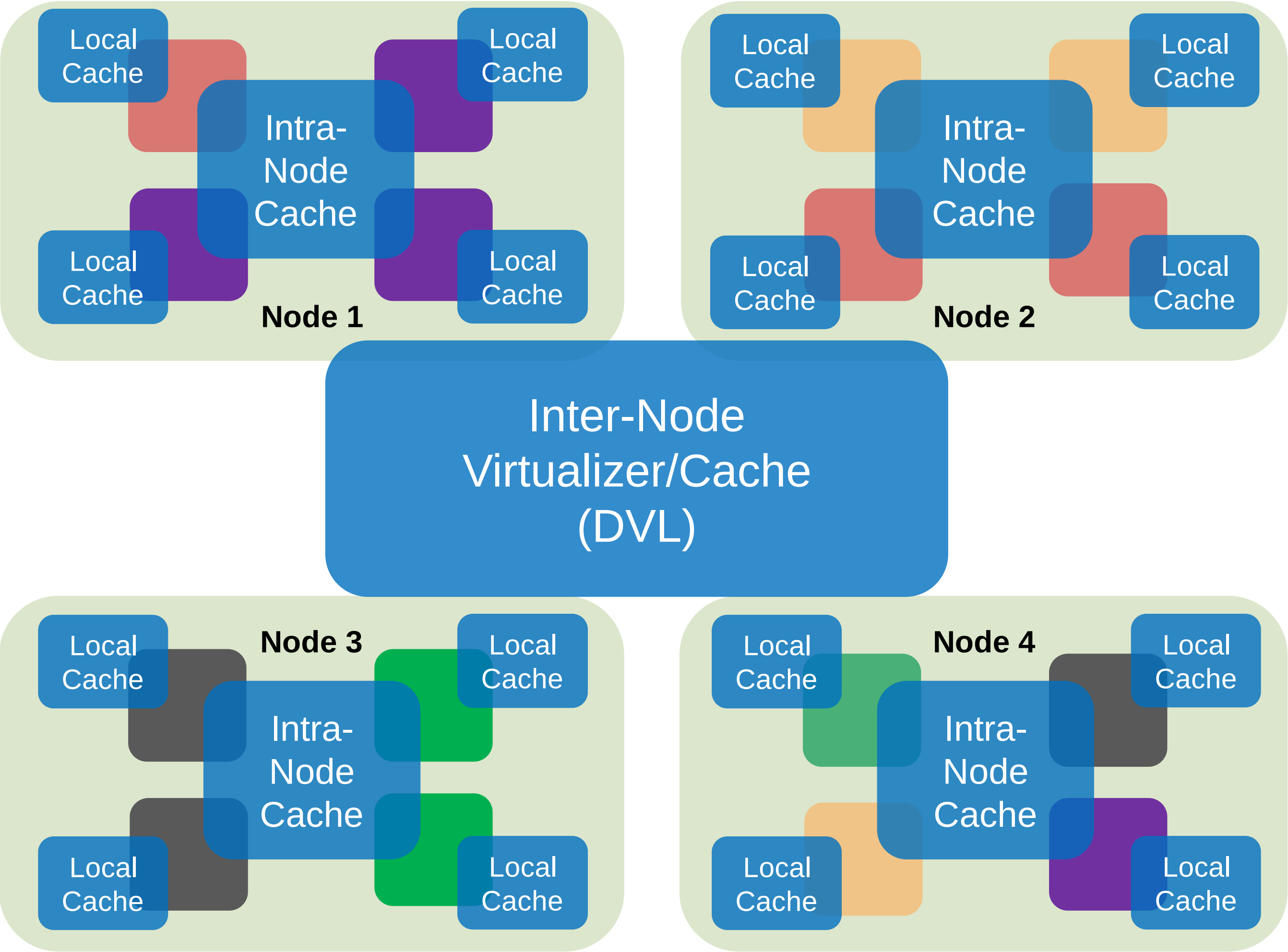
- ✗ Elasticity
- ✗ Persistent data
- ✓ I/O Capacity
- ✓ I/O Bandwidth

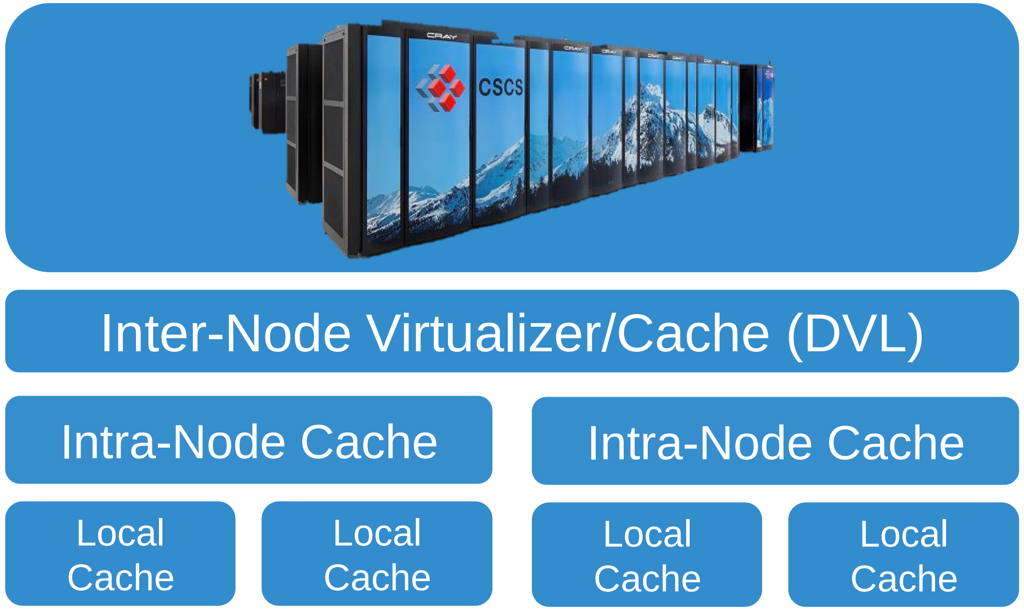
Analysis/Visualization Tools

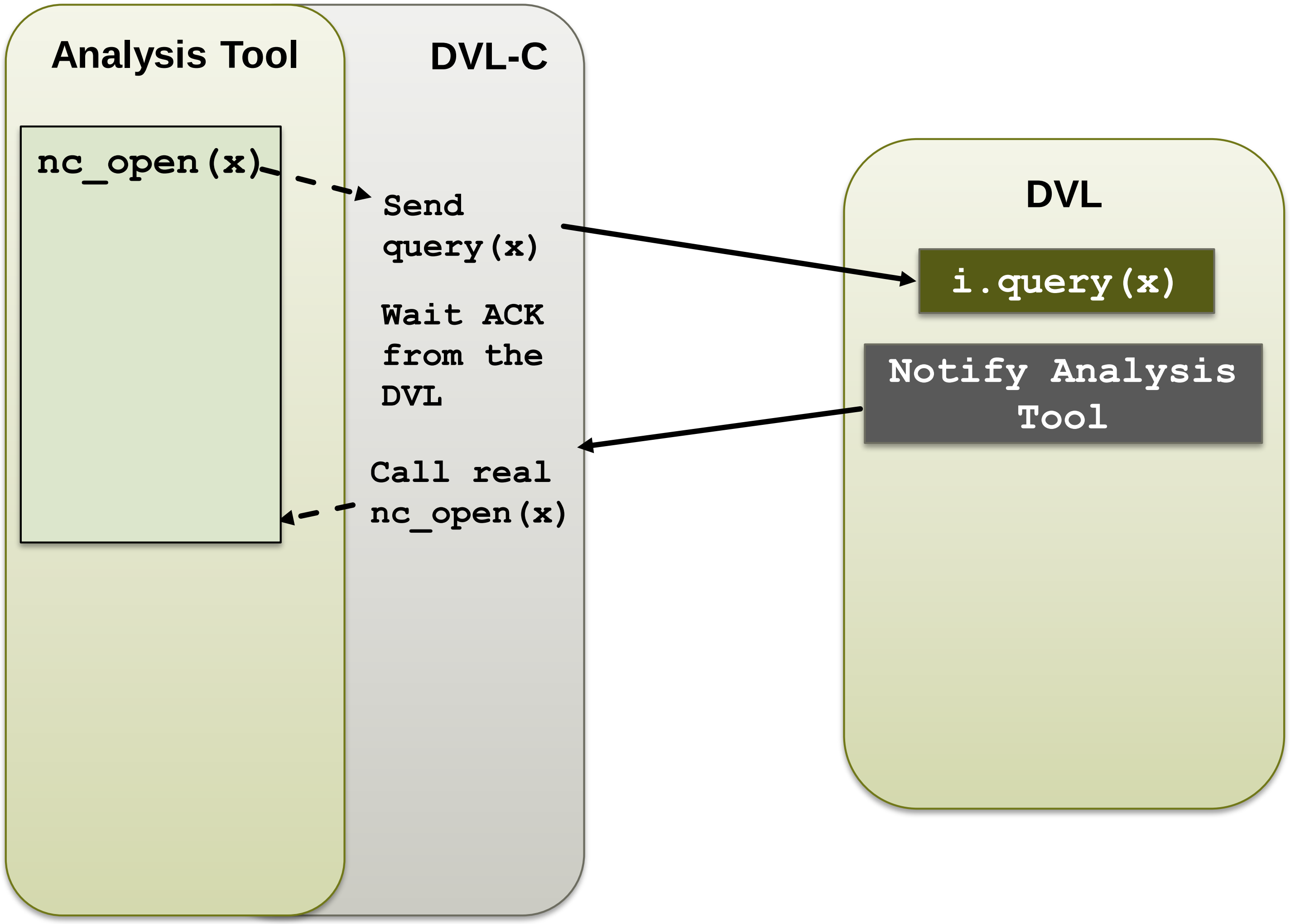
T1	T3
T2	T4

2 Get simulation data*

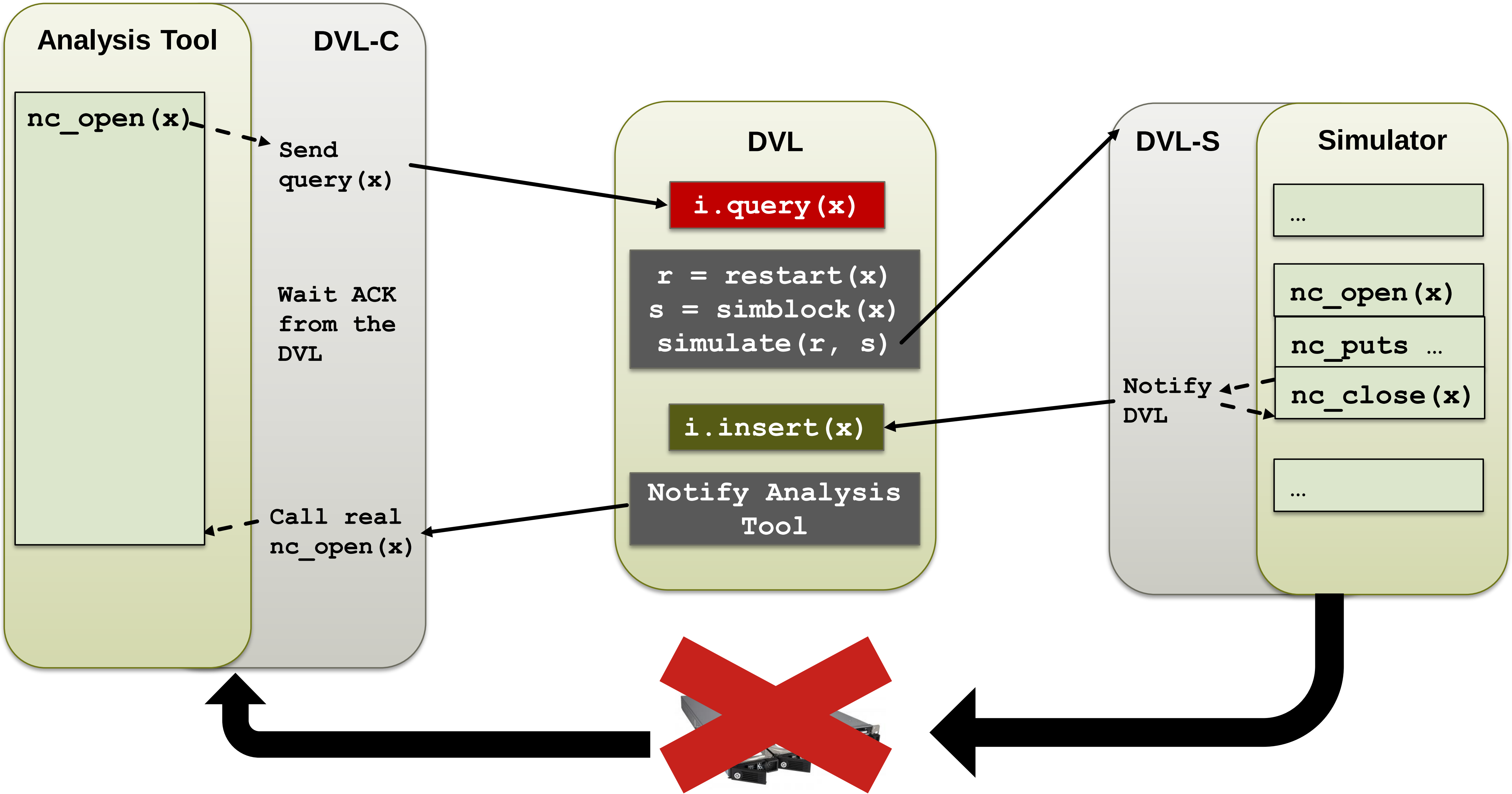
SDaVI Framework

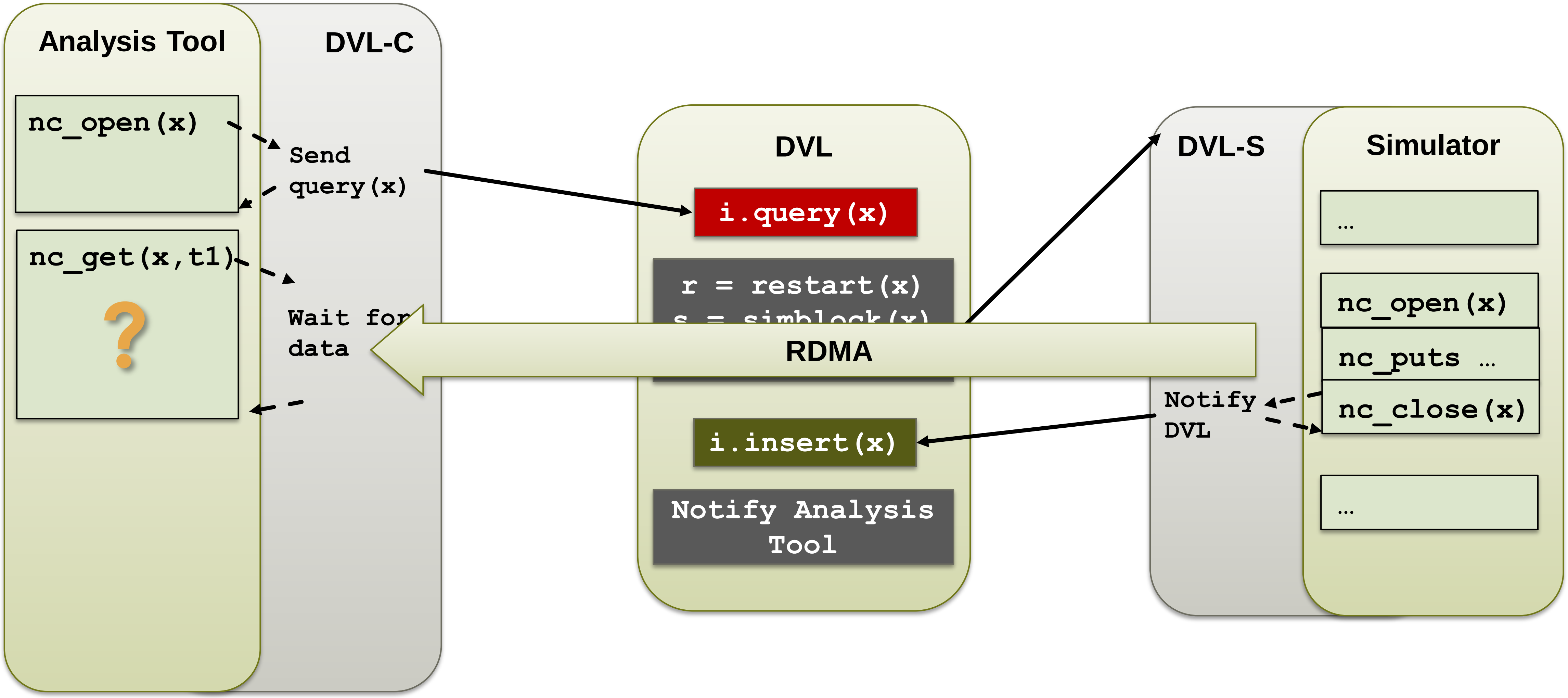






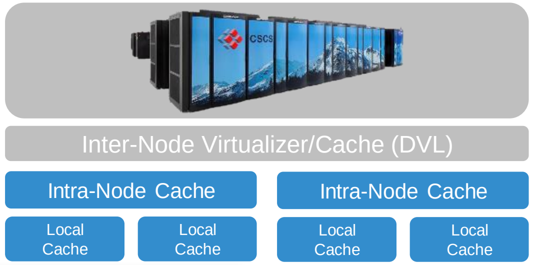
Hit = Offline Simulation

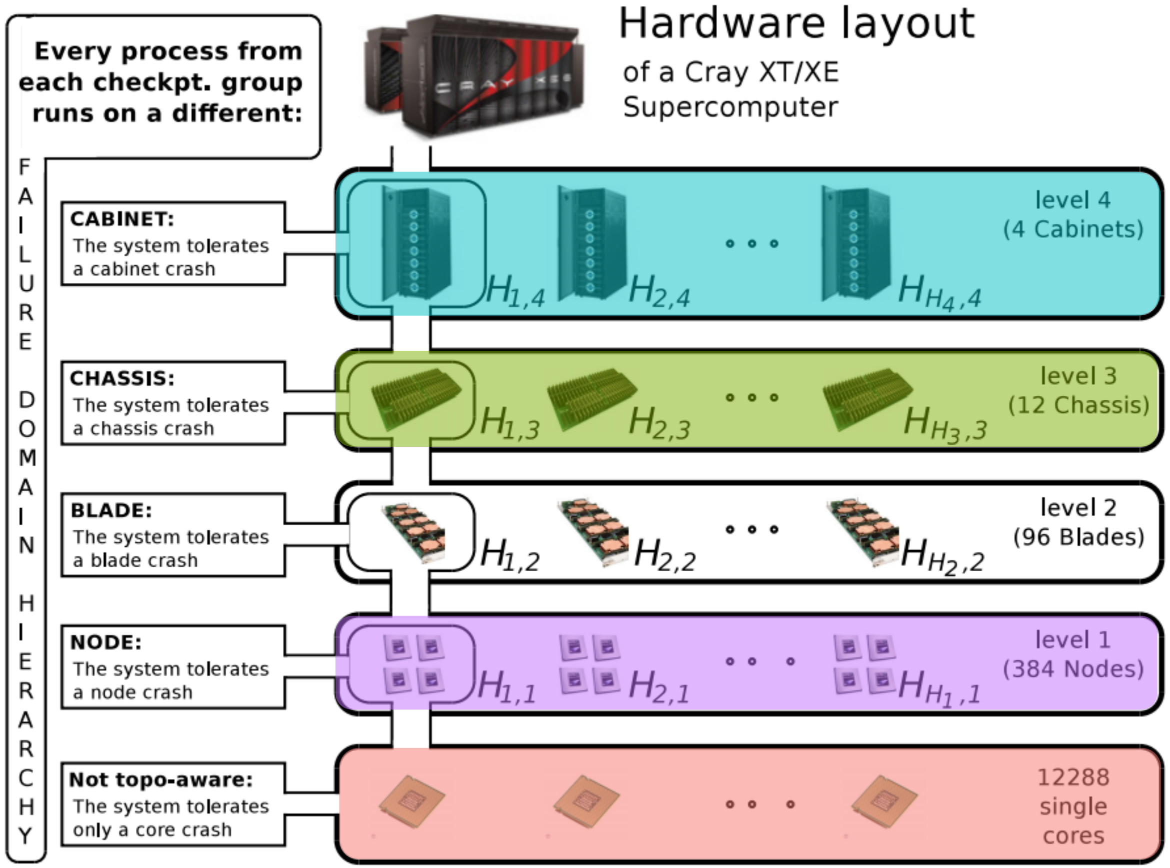

Analysis Tool
DVL-C
nc_open(x)
Send
query(x)
Wait ACK
from the
DVL
Call real
nc_open(x)
DVL
i.query(x)
r = restart(x)
s = simblock(x)
simulate(r, s)
i.insert(x)
Notify Analysis
Tool
DVL-S
Notify
DVL
Simulator
...
nc_open(x)
nc_puts ...
nc_close(x)
...



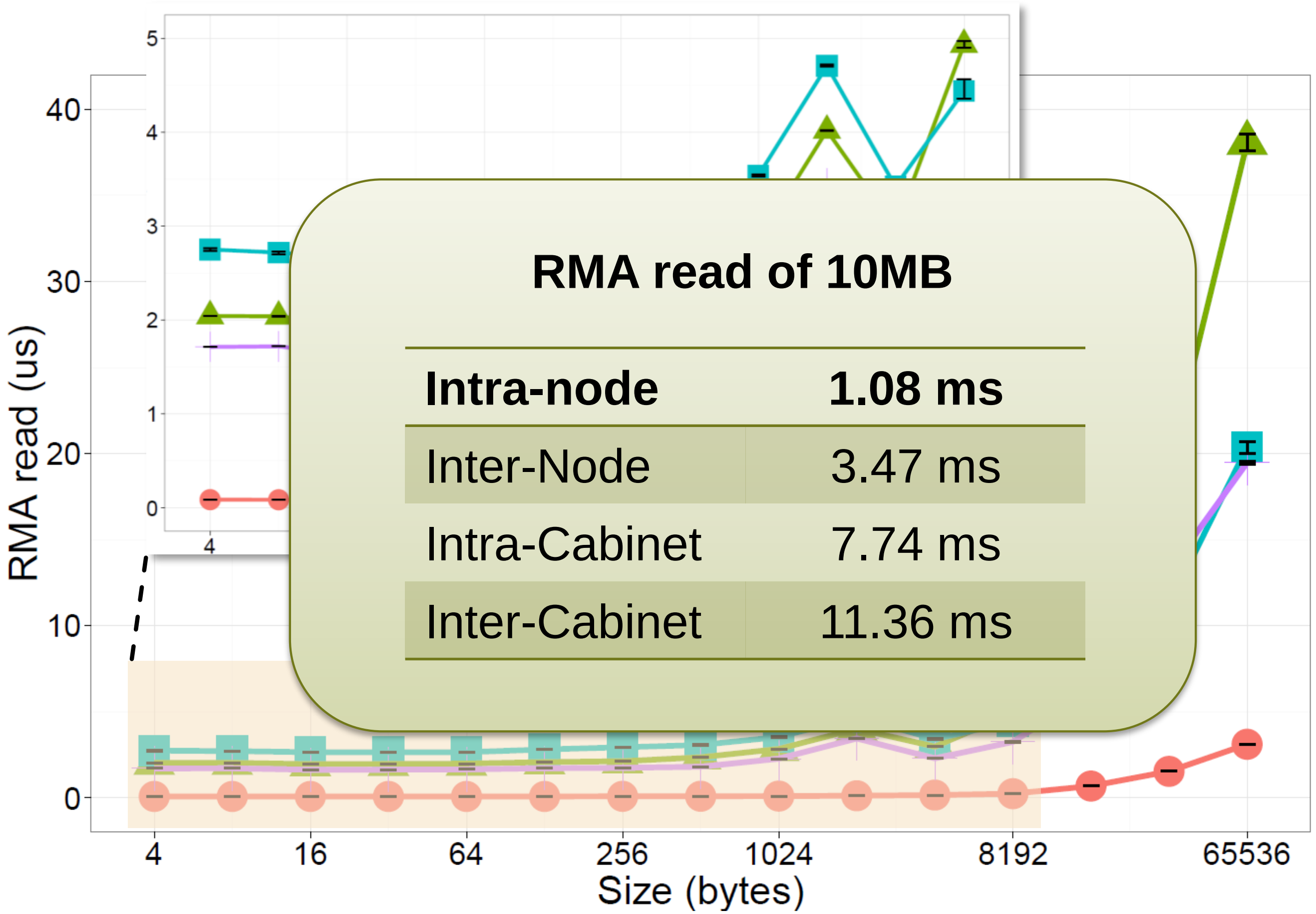
Miss = In Situ Simulation

Does intra/local node caching make sense?





M. Besta, T. Hoefler, Fault Tolerance for Remote Memory Access Programming Models, HPDC'14



RMA read of 10MB

Intra-node	**1.08 ms**
Inter-Node	3.47 ms
Intra-Cabinet	7.74 ms
Inter-Cabinet	11.36 ms

Establishing the IO-500 Benchmark

Julian M. Kunkel, John Bent, Jay Lofstead, George S. Markomanolis

2017-11-13

http://www.io500.org

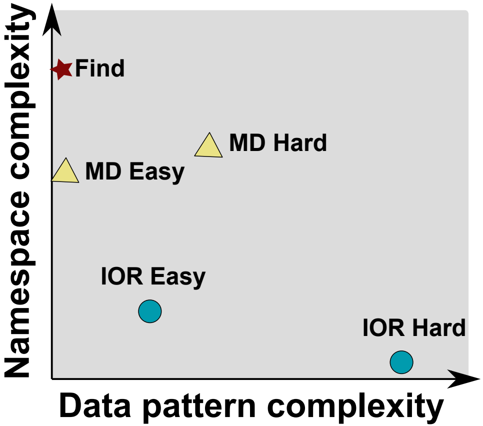
The IO-500

Goals

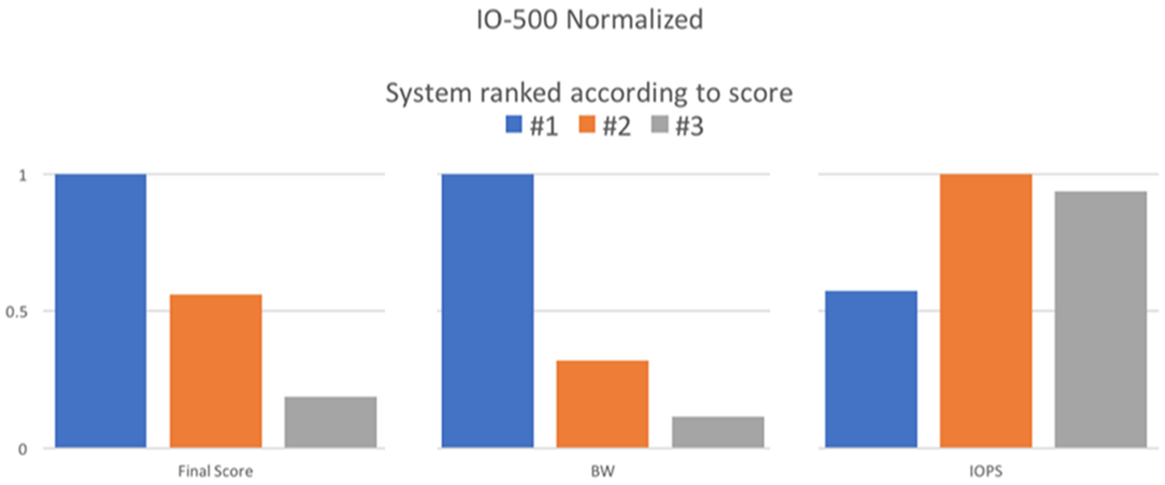
- Tracking storage performance
- Sharing best practices

Benchmarking Approach

- Community driven effort
- Patterns: metadata, data, search
  - Easy for optimized patterns
  - Hard for naive patterns
- Relies on community benchmarks



List Results from BeeGFS, DataWarp, IME, Spectrum Scale, Lustre



Challenges of Establishing the Benchmark

This is a short summary of experience gained by

- Feedback from discussions
  - From SC/ISC BoFs
  - Peers
- Feedback of people executing the IO-500 on different systems
- *Thanks to everybody contributing*

Challenges & Approach

Representative of applications and user requirements

- Supply workloads providing
  - Upper bound for optimized applications
  - Performance expectation for non-optimized applications
- *More workloads and concurrent execution to be integrated*

Understandable and human comprehensive results

- Report meaningful metrics
- Imply low variability of repeated measurements
- Computing of an overall score for ranking but retain individual values

Challenges & Approach

Portable

- Ran into Python (Shell) portability issues
- C-APIs: `readdir()` does not return type on DataWarp
- Non-POSIX `stat()` call on one system

Inclusive: cover various storage technology and non-POSIX APIs

- Allow vendors to use specific optimizations (for easy runs)
  - Enable replacement for find (IBM Spectrum Scale has optimizations here)
- Relying on (IOR's) AIOR interface (thanks to Nathan for porting mdtest)
- *We are still the process to support more storage APIs*

Challenges & Approach

Scalable, i.e., run on large-scale computers and relevant storage systems

- IOR and mdtest are MPI parallelized
- Supply a parallel find version

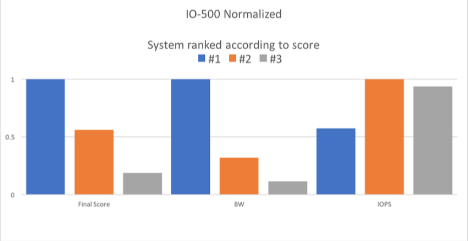
Lightweight: easy to setup and cheap to run

- 5 minute write/creation phases to limit runtime
- Extended IOR/mdtest for phase-out stonewalling options

Trustworthy: prevent (unintended) cheating

- Reveal all tunings made (also shares best practice)
- Sufficiently large working set

Visit our Birds of a Feather at SC



IO500

Getting Stared with IO500

- git clone https://github.com/VI4IO/io-500-dev
- cd io-500-dev
- ./utilities/prepare.sh
- ./io500.sh
- # Tune and rerun until good
- # email results to submit@io500.org

Contact us

http://www.io500.org
Slack: vi4io.slack.com
Twitter: IO500benchmark

Come see the full IO-500 results at SC17 BOF
Wednesday, 15 November, 17:15, room 201-203

"Results from ThinkParQ BeeGFS, Cray DataWarp, DDN IME, IBM Spectrum Scale, and Lustre!"