SCIENCE-DRIVEN DATA MANAGEMENT FOR MULTI-TIERED STORAGE

June 23, 2016

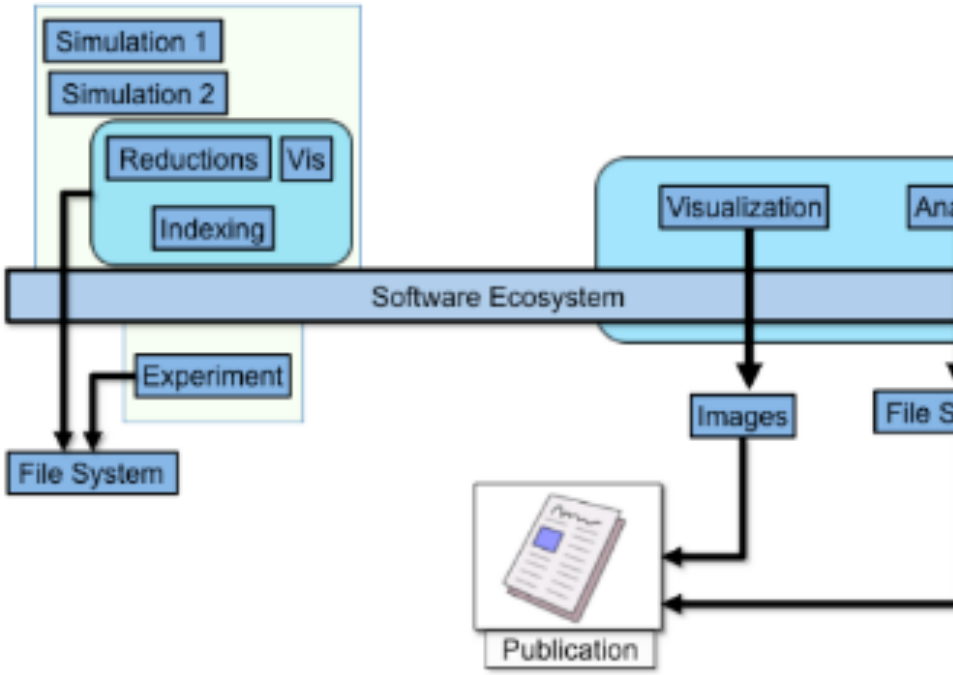
HPC-IODC Workshop

ay Lofstead	Sandia
Klasky, H. Abbasi, Q. Liu, F. Wang	ORNL
Lofstead, M. Curry, L. Ward	Sandia
I. Parashar	Rutgers
. Maltzahn	UCSC
I. Ainsworth	Brown, ORNL

Where Do We Spend Our Time in Science?

Goals

- Make the process **predictable**
- Make the process **adaptable**
- Make the process **scalable**
- Make the software **easy-to-use**



Observation

- Too much time is spent in managing, moving, storing, retrieving , and turning th science data into **knowledge**

Sirius specific Goal

- Refactor data so that we can efficiently store, find, retrieve data in predictable times set by the users, negotiated with the system

SIRIUS Story

xascale Apps will generate too much data

- Data needs to be prioritized by the users generating and reading
- Data needs to be reduced, re-organized, from large volumes into different buckets of importance
- Storage systems may not have the capacity or performance to sustain large datasets

he management of data in multi-tier Storage systems in the data lifecycle w
ecome more difficult to manage

ata access times need to be described by users and negotiated with the
torage system for predictable storage performance

hallenges:

- How can the middleware and storage system understand refactored data?
- How to build scalable metadata searching to find refactored data
- The interplay between data refactoring, data re-generation, and data access tim

Principles

rinciple 1: A knowledge-centric system design that allows user nowledge to define data policies

- Today SSIO layers are written in a stove-pipe fashion, and quite often do not all optimizations to take place
- Re-design the layers in a highly integrated fashion where users place their intentions into the system and actions will statically and dynamically take place optimize for the system and for individual requests

rinciple 2: Predictable performance and quality of data in the SSIO ayers need to be established so science can be done on the exascale ystems in a more efficient manner

- Without predictable performance, not only can the runs be slowed down becau of contentions on shared resources, but also it affects key science decisions

Outline

- Motivation
- IRIUS Building Blocks
- ata Refactoring
- uditing
- ata Description
- Metadata Searching
- uzzy Predictable Performance

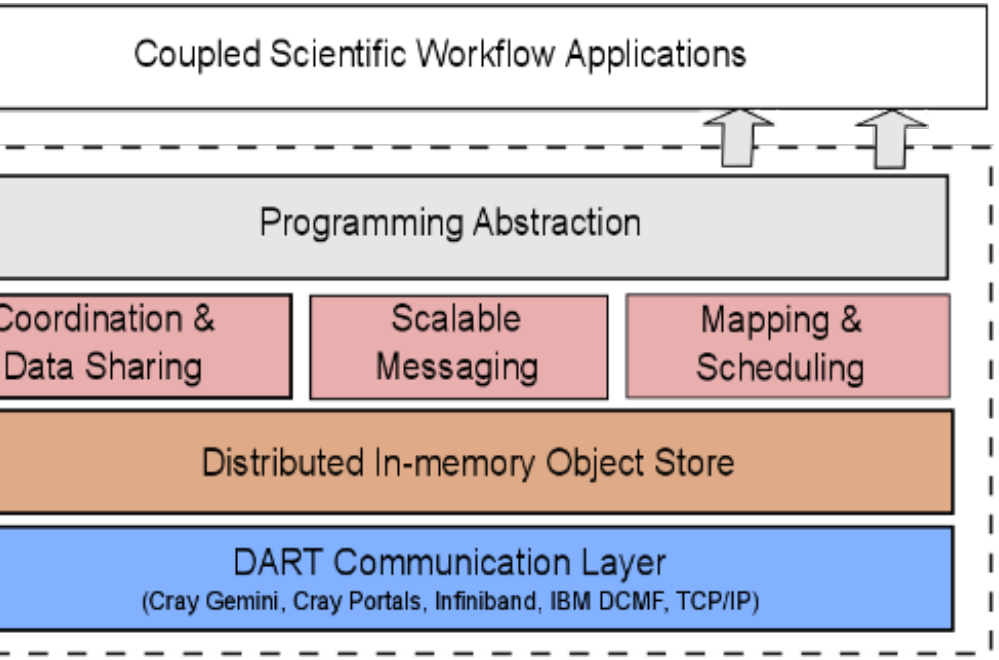
ADIOS – A critical library for DOE apps

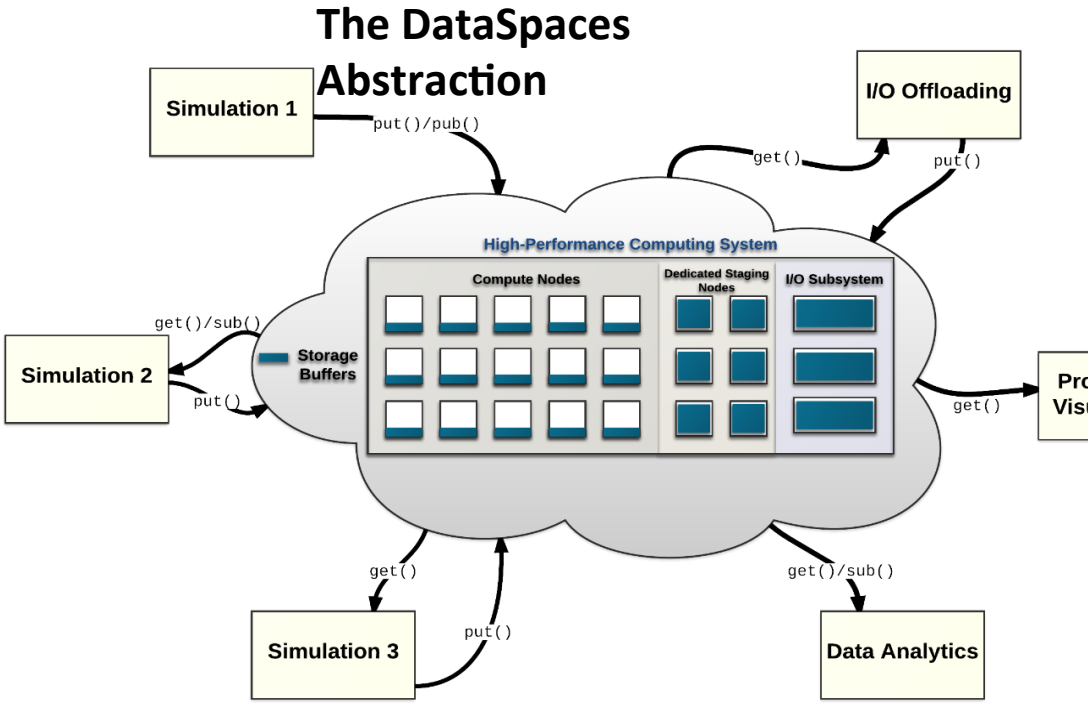
sed heavily in many LCF/NERSC applications which we partner with
llows our team to rapidly prototype new methods and test them for
pplication data lifecycles
as the ability to describe data utility (XML description)
llows our team to test out hypothesis

1. Accelerator: PIConGPU, Warp
2. Astrophysics: **Chimera**
3. Combustion: **S3D**
4. CFD: FINE/Turbo, OpenFoam
5. Fusion: **XGC**, **GTC**, GTC-P, M3D,M3D-C1, M3D-K, Pixie3D
6. Geoscience: **SPECFEM3D_GLOBE**, AWP-ODC, RTM
7. Materials Science: **QMCPack**
8. Medical: **Cancer pathology imaging**
9. Quantum Turbulence: QLG2Q
10. Visualization: Paraview, Visit, VTK, OpenCV, VTKm

://www.nccs.gov/user-support/center-projects/adios/

DataSpaces to explore placement strategies on multi-tier memory/storage hierarchies





lows our team to stage data across memory and storage hierarchies with different data acement strategies

ilding on our "learning" techniques to optimize data placement for different optimizatio rategies

Object Storage

Light Weight File System-inspired philosophy

- Clients bring/opt-in to services they require
- Naming, locking, distributed transactions

Peer-to-peer inspired design

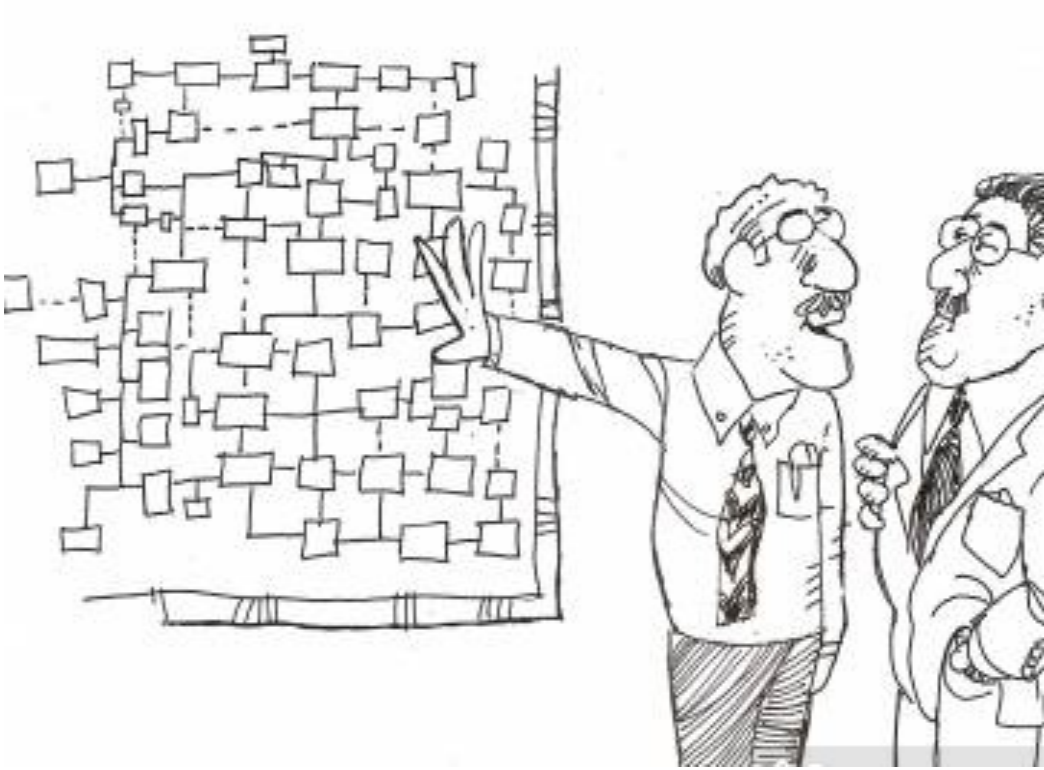
- Ephemeral, diverse servers and clients
- Data and location(s) are decoupled

Addressed systems

- Sirocco (Sandia) http://www.cs.sandia.gov/Scalable_IO/sirocco/
  - Full control from ownership, in development technology
- Ceph (RedHat)
  - Mature, production system with special attention required to address HPC workloads
- DAOS (Intel)
  - Next generation Lustre with strong commercial and DOE support, still under development

Outline

- Motivation
- SIRIUS Building Blocks
- Data Refactoring
- Auditing
- Data Description
- Metadata Searching
- Fuzzy Predictable Performance



Data Refactoring

ata needs to be refactored so that more important data can be rioritized, e.g., being placed on the higher tier on the storage systems.

hallenges:

- To understand the cost associated with refactoring data, in terms of CPU cycles, extra memory consumed, and communication.
- Does the refactoring affect the fidelity of the applications? If so, how much?
- After data is refactored, how do we map it to the storage hierarchy? How do we enforce policy?
- How will data be used? E.g.
  - B is a fundamental variable in a MHD code, but many times the user want the current: $J = \nabla \times B$ often looking at the low frequency modes, or $\nabla \cdot B$
  - We need error bounds for our refactored quantities

Data Refactoring and Utility Functions

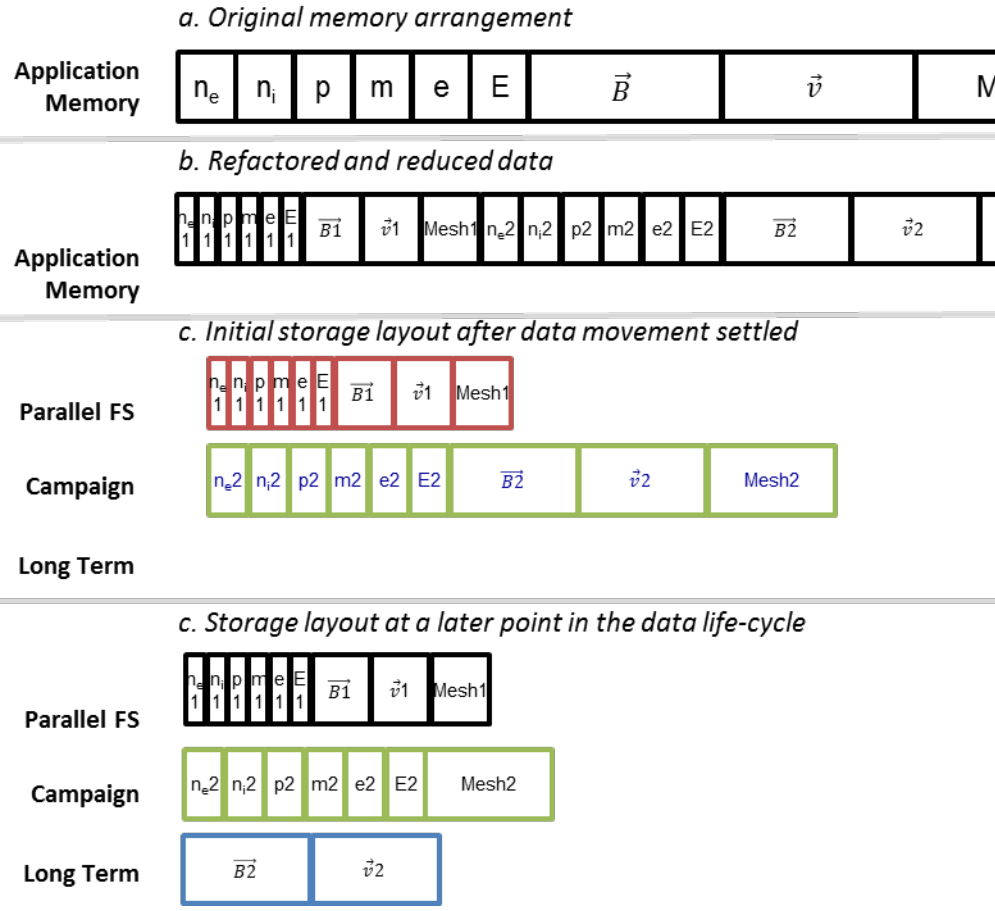
Need to explore what and where the computation will occur

- Flexibility in location based on past work
- Flexibility in which operation to perform on which chunk of data

Code generation or code containers are potential study targets

Maintaining relationship between data chunks

Carry attributes from generation to consumption and feedback into a utility computation



Data Utility

Describes how long a data chunk will live at a level of the storage hierarchy

Utility is a broad description

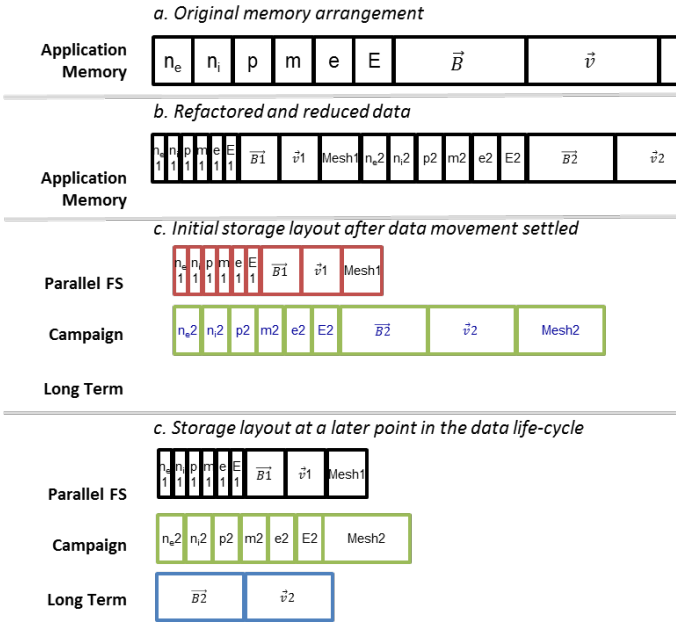
- Spatial or temporal utility of data
- Utility based on in-data features
- Utility based on statistical features

Utility has a large component from the user and the use case

- Experimental design factors in here
- Solving a specific scientific problem => specific data utility function

API for ingesting user preferences and combining with historical provenance

Dynamic utility for online analysis/visualization use cases



e.g. The utility of Mesh may defined more explicitly as, fo example, (priority=1, (time-NVRAM=8 hours, time-PFS=3 days, time-CAMPAIGN=100 days, time-TAPE=1000 days), (priority=2, (time-NVRAM=1 hours, time-PFS=4 days, time CAMPAIGN=100 days, time-TAPE=300 days).

Utility-driven Data Placement

oal: Determine placement of data objects **vertically** across different levels of he memory hierarchy, e.g., SSD or DRAM, and **horizontally** at different staging odes

- Utility quantifies the relative value of data objects based on anticipated data read patterns
  - Utility based on data access patterns (monitored and learnt at runtime) and the location of the application and staging nodes within the system network topology
  - For example, data objects with ***higher data utility*** are placed ***closer*** to the computing nodes accessing it

Exploring Data Staging Across Deep Memory Hierarchies for Coupled Data Intensive Simulation Workflows. T. Jin, F. Zhang, Q. Sun, H. Bui, M. Romanus, N. Podhorszki, S. Klasky, H. Kolla, J. Chen, R. Hager, C. Chang, M. Parashar. *IEEE* IPDPS'15, May 2015

Adaptive Data Placement For Staging-Based Coupled Scientific Workflows. Q. Sun, T. Jin, M. Romanus, H. Bui, F. Zhang, H. Yu, H. Kolla, S. Klasky, J. Chen, M. Parashar. ACM/IEEE SC'15, Nov. 2015.

Performance

nitial tests use 3/5 byte splits for doubles

- XGC particle data -- wrote 819,200,000 particles using 5 nodes (160 processes) o Sith @ ORNL
- Write with no compression :: 10.3s
- The time to split each double from that set of particles into 3 significant and 5 le significant bytes. :: 4.1s
- The time to write out the 3 byte pieces :: 3.4s
- The time to write out the 5 byte pieces :: 9.3s

eparate read times on a laptop

- Errors: Norms L2: 72028.2 Linf: 0.00109242
- Total ReadTime: 10.3131 decompressTime: 27.9715
- Whole data ReadTime: 34.1505 decompressTime: 0

Results sets

ariety of reading options

#	Size	Time	Time Err	Refactoring	Data Err
1	$(\frac{1}{4})^3(\frac{3}{8})$A	10s	$\pm$ 3s	stride, byte-split	99%
2	$(\frac{3}{4})^3(\frac{3}{8})$A	90s	$\pm$ 30s	stride, byte-split	58%
3	$(\frac{1}{4})^3(\frac{5}{8})$A	16s	$\pm$ 5s	stride, byte-split	0.01%
4	$(\frac{3}{4})^3(\frac{5}{8})$A	120s	$\pm$ 50s	stride, byte-split	0.01%
5	$(\frac{1}{4})^3$A	1200s	$\pm$ 30s	stride	98%
6	$(\frac{3}{4})^3$A	2400s	$\pm$ 90s	stride	58%
7	$(\frac{3}{8})$A	1350s	$\pm$ 120s	byte-split	5%
8	$(\frac{5}{8})$A	2250s	$\pm$ 120s	byte-split	0.01%
9	A	36s	$\pm$ 6s	wavelet	1%
10	A	3600s	$\pm$ 600s	none	0%

Outline

- Motivation
- SIRIUS Building Blocks
- Data Refactoring
- Auditing
- Data Description
- Metadata Searching
- Fuzzy Predictable Performance

New techniques for "Data Intensive Science"

UDITOR: Creating a reduced model to approximate the solution in
ited spatial/temporal dimenion.

sic Idea is that we need a model to generate a close approximatio
the data

sic quantities in Information Theory

ata stream S and for x ∈ S let $P_r(X=x) = p_x \in [0,1]$

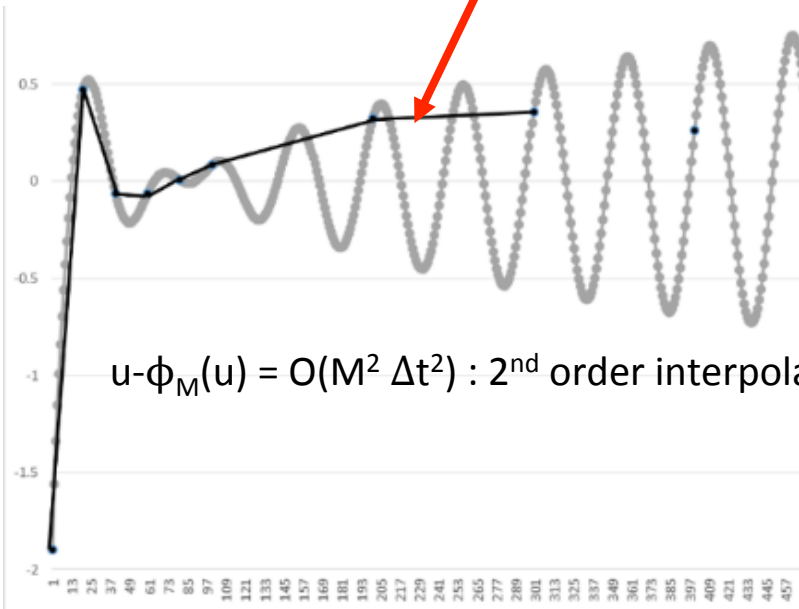
hannon Information Content: $h(x) = - \log_2 p_x$

ntropy $H(S) = - \Sigma\, p_x \log_2 p_x$

loisy/random data has HIGH ENTROPY

Current practices of today

Linear interpolation



Want to write data every m^{th} timestep

- Because of the Storage and I/O requirements users are forced to writing less

the users reconstruct their data, u(t), at he n^{th} timestep, they need to interpolate etween the neighboring timesteps

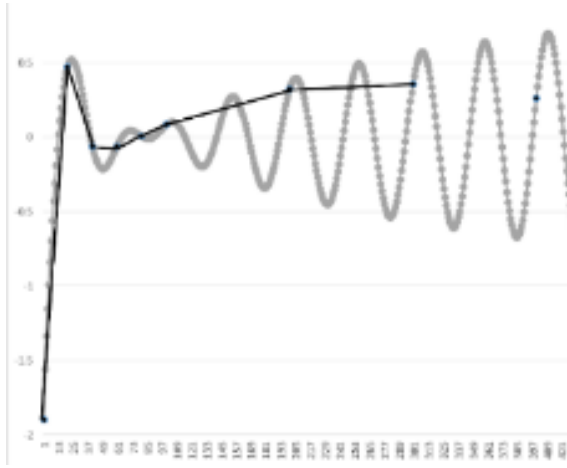
- $\Phi_M(u)$ = interpolant on coarser grid (stride M), reduce storage my 1/M

ssume (C=constant depending on the complexity of the data)

- Original storage cost = 32*N bits (floats)
- New storage cost = 32*N/M bits + { 23 – $\log_2$ (C M^2 Δt^2)}N
- Ratio = (1/M – 1/16 $\log_2$ M) – 1/16 $\log_2$ Δt + constant

Cost to store ϕ_M +
Cost to store mantissa of u- c

Compression with an interpolation auditor



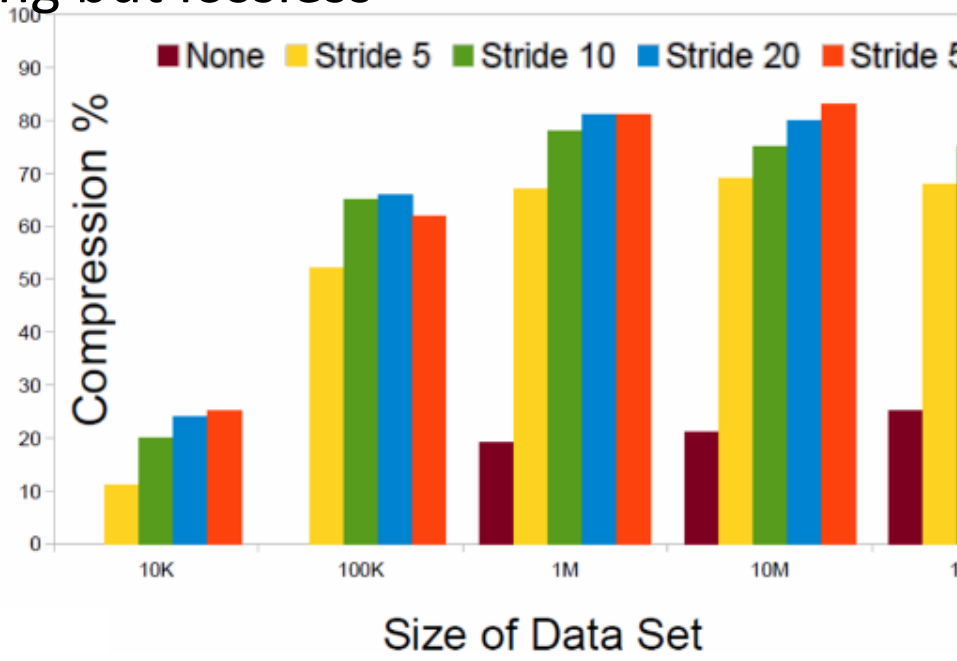
Linear interpolation (LA) is the auditor

If we look at 10MB output, with a stride of 5

- Total output = 50MB for 5 steps
- 10 MB, if we output 1 step, 43MB "typical lossless compression", 18MB, using linear auditing but lossless

Investigating adaptive techniques

	1 step (MB)	lossless compression (MB)	Linear Audit (MB)	Total Data in 50 steps, typical compression	**Total data in 50 steps in LA**
	10	43	18	430	**180**
	10	85	25	850	**125**
	10	170	40	1700	**100**



Other types of auditors

The key to better auditors is to understand what you are simulating/ observing

- Use a reduce model
- Use less resolution
- Use a linear equation for short spatial/temporal regions

And other ways to refactor

- Precision based re-organization
- Frequency based re-organization - Wavelets
- More knowledgeable auditors
  - Cost of data re-generation vs. data storage/ retrieval

Storage becomes more than a stream of bytes

- Data + Code + workflow

Outline

- Motivation
- SIRIUS Building Blocks
- Data Refactoring
- Auditing
- Data Description
- Metadata Searching
- Fuzzy Predictable Performance

Data Descriptions

ow that data is refactored, how do we describe/annotate data so that
ney can be discovered?

o capture application knowledge and communicate them to
niddleware/storage

- **Data utility**
- **Relationships between datasets**
- Semantics

o specify user requirements

- QoS: bandwidth, latency
- Policy: E.g., where and how long should my data stay on a storage layer

Outline

- Motivation
- SIRIUS Building Blocks
- Data Refactoring
- Auditing
- Data Description
- Metadata Searching
- Fuzzy Predictable Performance

Challenges in metadata searching

torage devices rather than a file system: no built-in metadata operations as art of IO

istributed pieces EVERYWHERE

esilience copies

torage devices come and go

erformance characteristics can vary considerably

ick a variety of storage targets based on data “importance”

ifferent data “compression” on different data pieces

ry to “guarantee” performance

- Need to consider decompression/regeneration time if multiple versions exist

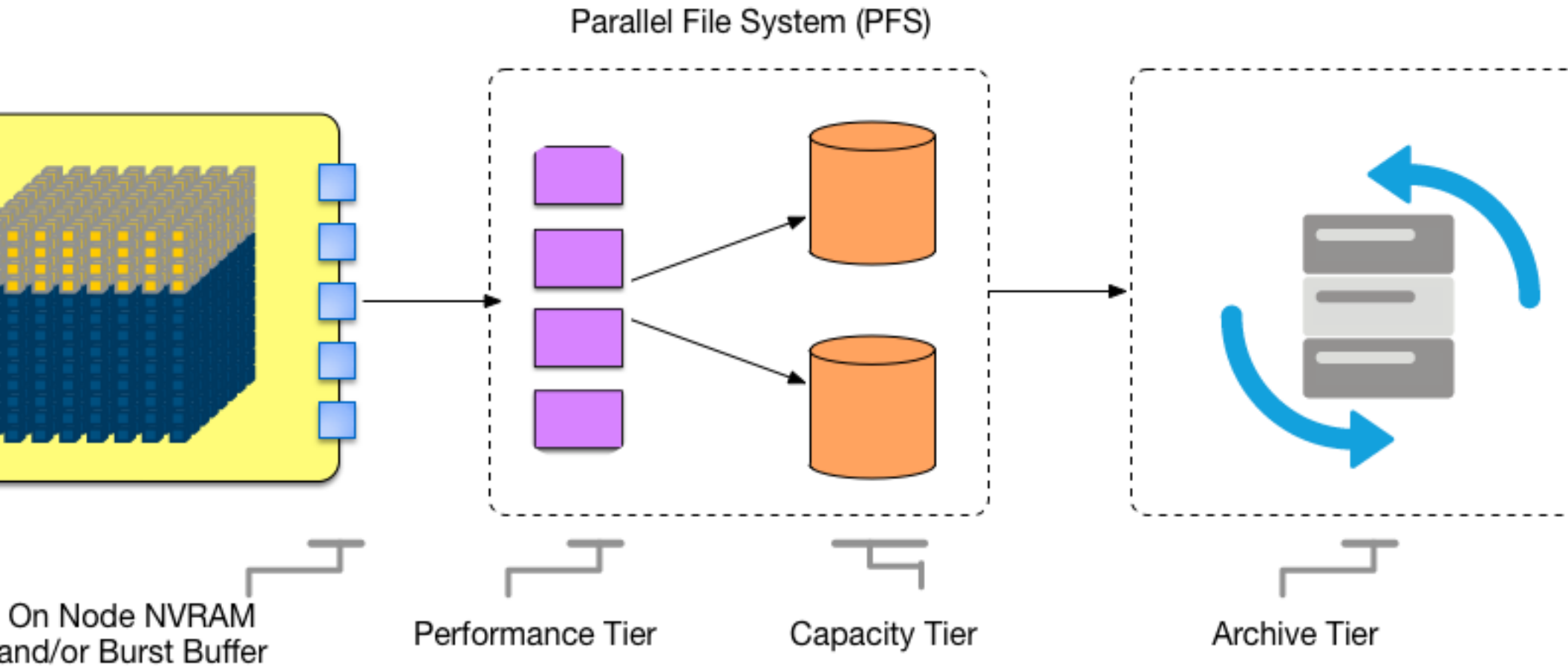
nhance placement decision based on predicted future use

- Based on tracking previous use (which needs to be tracked somehow)

Outline

- Motivation
- SIRIUS Building Blocks
- Data Refactoring
- Auditing
- Data Description
- Metadata Searching
- Fuzzy Predictable Performance

Current day end-to-end path is complex



Autonomic Runtime Optimization

utonomic Objective (AO):

A requirement/objective/goal defined by the user

E.g. minimize data movement, optimize throughput, etc.

utonomic Policy (AP)

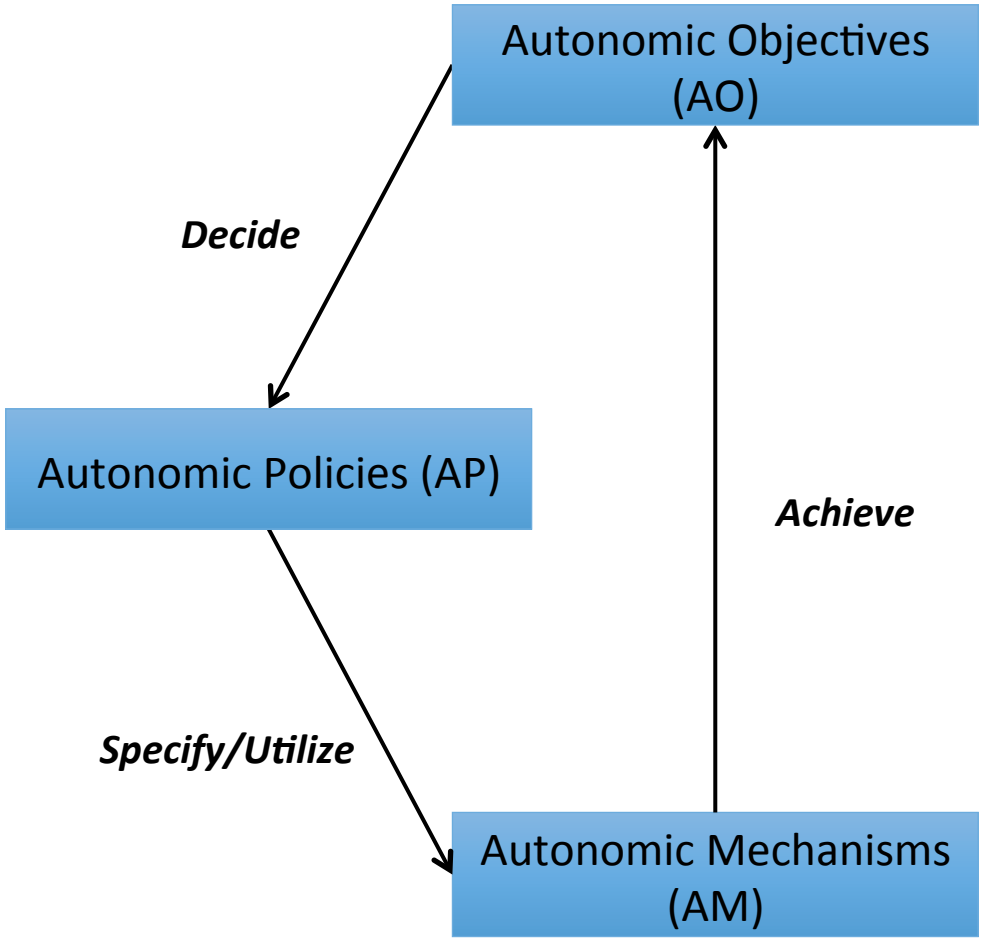
A rule that defines how the objectives should be achieved, i.e., which mechanisms should be used

E.g. use a specific data placement adaptation to minimize data movement , etc.

utonomic Mechanism (AM)

An action that can be used to achieve an AO

E.g. use topology-aware and access-pattern driven data placement to minimize data movement, etc.



Challenges

calable admission control

- Need scalable control plane (leverage existing work on causal metadata propagation and online sampling for latency/error trade-offs)

esource-specific schedulers to make performance of resource redictable

- Example: read/write separation at flash devices

nline sampling to provide quick latency/resolution trade-offs

- Without significantly interfering with ongoing workload

Summary

irius is attempting to redefine I/O based on key findings

- POSIX-compliant block interface does not give the system enough information t fully optimize
- Data is too big to keep in one place and current systems purge data without use intervention
- Variability is too large, and users are not in control of their data

cientific Data is not random data

- There is content to the data

uditing calculations to prioritize, reduce data sizes but keep the nformation is critical to reduce the time of understanding