# Tango: A Cross-layer Approach to Managing I/O Interference over Local Ephemeral Storage

Zhenbo Qiao
New Jersey Institute of Technology
Newark, NJ, USA
qjpqzb@gmail.com

Qirui Tian
New Jersey Institute of Technology
Newark, NJ, USA
qt2@njit.edu

Zhenlu Qin
New Jersey Institute of Technology
Newark, NJ, USA
zq53@njit.edu

Jinzhen Wang
City University of New York - Brooklyn College
New York City, NY, USA
jinzhen.wang@brooklyn.cuny.edu

Qing Liu
New Jersey Institute of Technology
Newark, NJ, USA
qing.liu@njit.edu

Norbert Podhorszki
Oak Ridge National Laboratory
Oak Ridge, TN, USA
pnorbert@ornl.gov

Scott Klasky
Oak Ridge National Laboratory
Oak Ridge, TN, USA
klasky@ornl.gov

Hongjian Zhu
X-Byte Research
Tenafly, NJ, USA
hzhu@xbyteresearch.com

***Abstract*—As simulation-based scientific discovery advances to exascale, a major question that the community is striving to answer is how to co-design data storage and complex physics-rich analytics in a way that the time to knowledge can be minimized for post-processing. A particular challenge is how to accommodate a broad spectrum of data analytics needs–particularly those that become clear only until very late during the post-processing, a scenario where existing methods, such as in situ processing, are unable or less effective in supporting data analytics. As HPC storage systems have become deeper and more complex with the recent addition of NVMe, die-stacked memory, and burst buffer, it requires fundamentally rethinking new paradigms and methods for data storage and analysis. This paper aims to address the issue of I/O interference for data analytics over local ephemeral storage, which is shared by multiple applications in a non-exclusive node usage scenario–often configured for small- to medium-sized clusters. At the core of this work is a coordinated cross-layer approach that reacts to storage interference from both storage and application layers. By decomposing and distributing analysis data across the storage hierarchy, data analytics can adapt to the interference by reducing or completely avoiding access to lower tiers whenever there is a high interference, while maintaining a prescribed error bound to limit the information loss. Meanwhile, proper actions are also taken at the storage layer to ensure sufficient bandwidth is allocated for retrieving an augmentation, which is based upon the cardinality and accuracy of the augmentation as well as the nature of an application. We evaluate three real-world data analytics, XGC, GenASiS, and CFD, on Chameleon, and quantitatively demonstrate that the I/O performance can be vastly improved, e.g., by 52% versus no adaptivity and 36% versus single-layer adaptivity, while maintaining acceptable outcomes of data analysis.**

***Index Terms*—High-performance computing, data analysis, data storage**

## I. Introduction

As simulation-based scientific discovery advances to exascale empowered by new accelerators and interconnects on high-performance computing (HPC) systems, the management of analysis data coming out of large-scale simulations has become increasingly challenging due to the exponential growth of data. Scientific knowledge discovery does not simply end after the completion of a simulation, but continues well into the post-processing phase where scientists repetitively retrieve and analyze data over a long period. A major question that the community is striving to answer is how to co-design data storage and complex data analytics in a way that the time to knowledge can be minimized in post-processing [1], [2], [3], [4], where either routine or exploratory data analytics (i.e., those that are unknown a priori) will be executed over large amounts of analysis output. A particular predicament is how to accommodate a broad spectrum of data analytics needs–particularly those that become clear only until very late during the post-processing, a scenario where existing methods, such as in situ processing [5], [6], [7], [8], storage layer optimization [9], [10], [11], and data reduction [12], [13], [14], [15], [2], are less effective in supporting data analytics. As HPC storage systems have become deeper and more complex with the recent addition of NVMe, die-stacked memory and burst buffer, it requires fundamentally rethinking new methods for data storage and analysis. While one can simply treat these new storage technologies as additional caching layers or faster file system directories, a more holistic approach calls for the hierarchical management of data, which is well suited to the structure and hardware characteristics of HPC storage systems. The key benefits of this new research direction are that 1)

SC24, November 17-22, 2024, Atlanta, Georgia, USA
979-8-3503-5291-7/24/$31.00 ©2024 IEEE

it allows a reduced representation of data to be placed and accessed from a fast storage tier, which otherwise would be too capacity-limited to store the entire dataset. As a result, data analytics can run much faster with vastly reduced I/O and compute overhead; 2) more importantly, it can satisfy a wide range of accuracy needs, thus being highly relevant to exploratory data analytics where the accuracy needs are unknown a priori–if the accuracy of a reduced presentation is insufficient, it can always be elevated by fetching additional augmentations from lower storage tiers.

The I/O performance on HPC systems is observed to be highly variable due to the lack of *end-to-end* quality of service (QoS) mechanisms in place [16], [17], [3] and the fact that HPC storage systems are generally shared by multiple applications. This is true for both remote and local storage systems. For the former, the storage system is connected through the network and shared among multiple nodes/systems, while for the latter, the storage system is directly attached to a node where several processes/threads can share the storage. The local storage system is mostly ephemeral in HPC–before a job starts, the analysis data needs to be staged into local storage, and after a job exists, the data on local storage is erased. While there has been a multitude of work focusing on reducing I/O interference on remote storage systems [17], [18], [19], the interference problem over local storage has rarely been studied for HPC workloads. As compared to remote storage, an application has more control of resources over local storage via Linux control groups (cgroups) typically available through containers, however, without achieving performance guarantees. Meanwhile, the concurrency of a compute node has improved drastically over the past decade. For example, a shared instance in a commercial cloud today could have hundreds of vCPUs [20], and a physical CPU could have up to 64 cores [21]. This creates more opportunities for interference over local storage.

**Target scenario.** This work targets a non-exclusive node usage scenario [22]–often configured for small- to medium-sized clusters [23], [24], [25] where multiple applications run on the same node to be resource efficient. As such, there will be I/O interference over the local ephemeral storage.

**Contribution.** This paper aims to address the issue of I/O interference for data analytics where the local ephemeral storage is shared by multiple applications. In particular, we developed Tango, which is a coordinated cross-layer approach that reacts to I/O interference from both storage and application layers. By decomposing and distributing analysis data across the storage hierarchy, data analytics can adapt to the interference by reducing the access to lower tiers whenever there is a high interference, here referred to as *application-layer adaptivity*, while maintaining a prescribed error bound to limit the information loss. Meanwhile, proper actions are taken at the storage layer to ensure sufficient bandwidth is allocated for retrieving an augmentation, here referred to as *storage-layer adaptivity*, based upon the cardinality and accuracy of the augmentation as well as the priority of an application. Inevitably, this design involves the estimation of storage performance, which can be done by exploiting the periodic behavior of HPC workloads [3]. Overall, this paper makes the following contributions.

- This work is among the first to explore a cross-layer approach to managing I/O interference over local ephemeral storage. In particular, we propose the idea that both applications and storage should adapt to best manage the interference. A single-layer adaptation in either storage [18], [19] or application [26], [3] would simply re-distribute the bandwidth between interfering applications without alleviating the contention, or suffer subpar storage performance. In this work, when the interference is predicted to be low, data analytics can retrieve more augmentations assisted by a higher allocation in the storage layer to achieve the best outcomes for data analysis. Meanwhile, when the interference is predicted to be high, data analytics should retrieve less (or no) augmentations and be allocated less storage resources.
- We develop new algorithms to achieve error-bounded refactorization and interference mitigation, where we introduce error control for normalized root mean square error (NRMSE) and peak signal-to-noise ratio (PSNR)– two widely used error metrics in data reduction, during both decomposition and recomposition. Further, we propose a weight function in the storage layer that balances the cardinality, the accuracy of an augmentation retrieved, and the priority of data analytics to best allocate storage resources.
- We experimentally demonstrate the effectiveness of the cross-layer approach on Chameleon [27] using real data analytics, XGC, GenASiS, and CFD. The results show that the I/O performance is vastly improved, e.g., by 52% versus no adaptivity and 36% versus single-layer adaptivity, while achieving acceptable analysis outcomes.

The remainder of this paper is organized as follows. Section II discusses the background and motivation of this work. Section III presents our cross-layer design of error-bounded refactorization and augmentation to manage I/O interference. Section IV demonstrates the performance advantages of the proposed scheme. Section V presents the most related works, along with conclusions in Section VI.

## II. Background and Motivation

**HPC storage QoS.** Table I describes the QoS mechanisms in major HPC file systems today. In particular, Lustre is a parallel file system that is being used by many large HPC systems today. Since Lustre 2.6, it has introduced a new feature that maps an RPC request to a QoS class based upon the client's network and job ID, and the type of I/O operation, and uses a token bucket filter to throttle the I/O traffic for a given class [28]. Nevertheless, the QoS control in Lustre is cumbersome and ad-hoc, since applications are not typically allocated with a static network and job ID, and cannot provide per-application performance guarantee. More importantly, the QoS control cannot adapt to the workload dynamic at runtime. Meanwhile, Spectrum Scale (5.0.4, formerly GPFS) provides I/O throttling for two QoS classes for each storage pool [29]. The QoS control is coarse-grained and enforced at each storage pool, rather than for each application, and cannot be adjusted unless the system administrators intervene to make manual changes. The Ceph distributed file system [30] has also gained traction in HPC centers and very recently, there have been efforts integrating *dmClock* [31] to provide a performance guarantee. However, we confirmed with the Ceph production team that the integration is still experimental and the performance guarantee has not been achieved as of now. Lastly, OrangeFS [32] is a file system in user space (FUSE) that is specifically designed to optimize MPI-IO in a parallel environment. OrangeFS itself does not provide QoS, but rather relies on the lower-level file system to achieve it.

*Motivation 1: Major HPC file systems do not provide application-level performance guarantees. This motivates us to seek alternative solutions at upper levels in the software stack, particularly at storage and I/O intermediaries, to manage I/O interference.*

**TABLE I:** QoS in HPC file systems.

| File system | Per app. control | Runtime adjust-ment | QoS mechanism | Scheduling |
|---|---|---|---|---|
| Lustre (>2.6) | ✗ | ✗ | Throttling | Token bucket filter |
| Spectrum Scale (5.0.4) | ✗ | ✗ | Throttling for two QoS classes for each storage pool | Unknown |
| Ceph (13.2.6) | ✗ | ✗ | Throttling | *dmclock* |
| OrangeFS (2.9.7) | ✗ | ✗ | ✗ | ✗ |
| Ext4 with cgroups | ✓ | ✓ | Proportional weight, throttling | Completely fair scheduling |

**Runtime resource control via cgroups.** Containers, and more broadly any systems with cgroups enabled, provide a range of mechanisms for resource control over local compute and storage. For example, the sharing of CPU cycles and block I/O (*blkio*) among containers can be adjusted proportionally via weight or be throttled at runtime by specifying the resource key-value pairs for the subsystem of a control group. For example, for block I/O, the proportional weight of a container can be specified by adjusting *blkio.weight* in the range from 100 to 1000. To set an upper limit of I/O operations, we can set the value of *blkio.throttle.read_bps_device* and *blkio.throttle.write_bps_device*.

However, the availability of proportional control through weight or throttling does not translate to performance isolation. In particular, the throttling function only limits the maximum I/O rate that a container can achieve, without guaranteeing the minimum rate–a metric that is of equal, if not more, importance to QoS. Meanwhile, a static weight can lead to a reduced bandwidth allocation when the interference becomes stronger. For example, setting an I/O weight of 100 for a target application will guarantee at least 50% of the bandwidth, when there is another interfering application with an equal weight. However, once a third interfering application with an equal weight joins, the target application will receive only 33% of the total bandwidth. As a result, the proportional allocation using cgroups does not solve the interference problem. In Fig. 1, we measure the average I/O performance (instead of the instantaneous bandwidth) of three data analytics, XGC, CFD, and GenASiS, which perform I/O periodically over time, on ext4 on a Seagate 600GB 15000 RPM SAS hard disk, with equal blkio weights. Clearly, the weight mechanism does not achieve performance isolation for these applications. For example, when XGC performs I/O without being interfered at $t = 10$, it almost achieves the full bandwidth from the underlying storage system. In contrast, when interfered at $t = 0$, the perceived bandwidth drops about 75%.

*Motivation 2: The weight assignment provides an important means in controlling local resource allocation. However, a static weight assignment in blkio is unable to achieve performance isolation from interference. This calls for a more sophisticated runtime management of weight.*

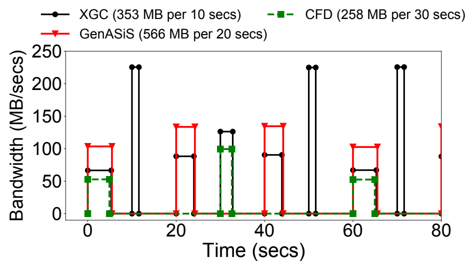
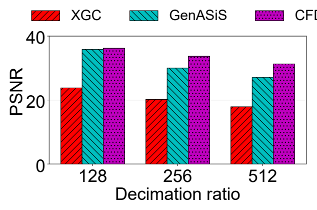


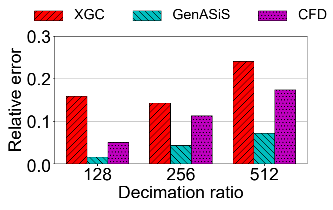
**Fig. 1:** I/O Performance of data analytics with equal weights on ext4 on a Seagate 600GB 15000 RPM SAS hard disk. Here we run three data analytics in different docker containers that retrieve and analyze data iteratively from the shared disk.

**Application-layer adaptivity.** It has been well recognized that HPC applications can adapt to computing environments in an ad-hoc fashion. The simplest form of such adaptations is that when computing resources are constrained, domain scientists often scale down the fidelity of the simulation by manually throwing out non-essential terms in the governing partial differential equations, coarsening the spatiotemporal resolution, or reducing the degrees of freedom. As a matter of fact, this was how production science was done when the HPC systems were much less powerful in the past, but still with meaningful results produced. Particularly, when the storage system is highly congested, applications can reduce the volume of data retrieved or the frequency of I/O operations. While such adaptations are not completely harmless and can discard important physics in data, it has been shown in the literature [3], [4] that the lower accuracy of data can still be useful for some data analysis, such as visualization and statistical analysis, especially for high-fidelity simulations where the solutions are over-resolved. More importantly, the lower accuracy can be instrumental in guiding the subsequent computation to be more localized, thus reducing the computational overhead. Fig. 2 illustrates PSNR and the relative error of the analysis outcome, when data is reduced. While it is clear that as the decimation ratio increases, the quality of data decreases, we do not observe drastically degraded analysis outcomes. On the other hand, even at an extreme decimation ratio of 512, a level of reduction that is rarely achieved, the relative error of data analytics is no more than 25%. Detailed descriptions of the data analytics are in Section IV-A.

*Motivation 3: While a reduced representation inevitably incurs errors to the outcome of data analytics, it can maintain a reasonable outcome and can be useful for data analysis. Nevertheless, error control mechanisms are needed to accommodate a wide range of accuracy needs.*



**(a)** PSNR.



**(b)** Analysis error.

**Fig. 2:** Accuracy of using a reduced representation.

**HPC application pattern.** Most parallel applications on HPC systems are deemed to follow certain patterns that can be described as $I(C^xW)^*F$ [33], [34]. In particular, $I$ and $F$ are the initialization and finalization phases, respectively. The compute phase $C$ and the I/O phase $W$ are typically iterative in nature, and a number of rounds ($x$) of compute iterations will be followed by an I/O phase, with $x$ being adjustable based upon the scientific needs as well as the performance of storage. By and large, the I/O interference can be viewed as the bandwidth competition from all other concurrent applications, denoted as $\sum_i I_i(C_i^xW_i)^*F_i$, where the subscript $i$ denotes the $i$-th competing application. As such, the interference pattern does exist and can be learned and predicted. Carefully note that other random activities, such as job compilation and shell commands, are much lower in their intensity and duration, as compared to the simulation output, and therefore can be neglected. In fact, prior work [3] shows the effectiveness of a discrete Fourier transform (DFT) approach where the random interference can be filtered out and the overall interference intensity and phase can be accurately estimated for a future analysis step.

## III. Design and Implementation

### A. Methodology

Fig. 3 provides a high-level view of Tango. For local ephemeral storage, we generally consider two kinds of storage tiers: performance tier (e.g., NVMe) and capacity tier (e.g., disks). The former has a relatively smaller capacity but a faster speed than the latter. The central idea is to manage the I/O interference through a coordinated cross-layer approach that involves both application and storage layers, as compared to prior works which mostly explored a single layer (see Table II for more details). In particular, at each layer, we perform the following actions:

- **Application layer.** The data analytics adapts to the interference by adjusting the accuracy of data representation based upon the estimated interference intensity at a given timestep $t_s$. In an ideal case when there is little interference on the capacity tier, a full augmentation would be retrieved, thus incurring no error to the outcome of data analytics. When there is a substantial interference, we proportionally reduce the amount of augmentation retrieved from the shared storage, to the extent that the prescribed error bound can be satisfied, so that the interference can be mitigated while the accuracy can be maintained as much as we can.
- **Storage layer.** The blkio weight of a particular application container will be adjusted at runtime accordingly, based upon the augmentation strategy constructed through performance monitoring and estimation, to ensure sufficient resources are allocated for a target application. In this work, we devise a weight function that maps the cardinality and the associated accuracy of an augmentation along with the priority of an application to blkio weight.

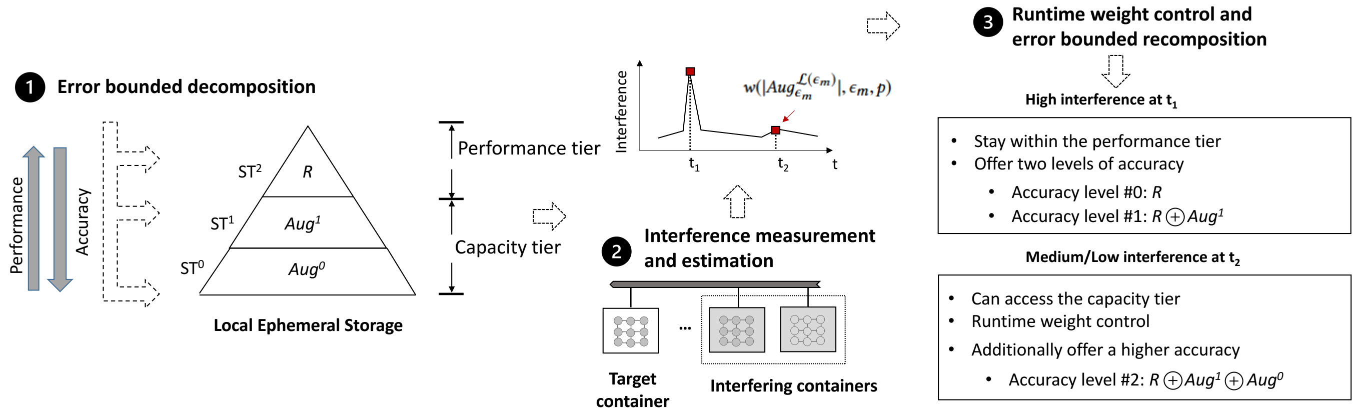
**TABLE II:** Comparison with existing methods.

| Work | Storage layer | App. layer | Technique |
|---|---|---|---|
| [18], [19] | ✓ | ✗ | Traffic re-routing and throttling based upon queue length |
| [17] | ✗ | ✓ | Explicit application coordination through new APIs |
| [26] | ✓ | ✗ | Randomized I/O scheduling |
| [3] | ✗ | ✓ | Interference estimation and adaptive data retrieval |
| [2] | ✗ | ✓ | Data retrieval under no interference |
| Tango | ✓ | ✓ | Cross-layer coordination involving storage- and application-layer adaptivity |

For explanatory purposes, notations used in this paper are listed in Table III. Consider a local storage hierarchy that consists of $L$ tiers $\{ST^l,\ 0 \le l < L\}$, with $ST^0$ being the slowest tier of the largest capacity and $ST^{L-1}$ being the fastest tier of the smallest capacity. An example of a three-tier storage is shown in Fig. 3.

First, to exploit such storage characteristics and be adaptable to I/O interference, the simulation output must be transformed in a way that enables a hierarchical representation $\{\Omega^l,\ 0 \le l < L\}$, where $\Omega^0$, $\Omega^l, 0 < l < L-1$, and $\Omega^{L-1}$ are the original, intermediate, and base representations, respectively. To reduce the storage footprint of these representations, we further compute the difference between adjacent levels of representations, i.e., $Aug^l$, which is the difference between $\Omega^l$ and $\Omega^{l+1}$. Furthermore, we transform $Aug^l$ in a way that a fine-grained set of error bounds can be achieved (step ❶) when a subset of $Aug^l$ is requested to be retrieved under interference, thus avoiding an excessive loss of information. Next, the decomposed data are staged to local ephemeral storage before a data analytics job starts, taking advantage of the capacity and speed characteristics of each tier.

Second, during data analysis that spans hundreds to thousands of steps, we collect the performance history of data analytics for the early steps and perform estimation for the



1 Error bounded decomposition
Performance
Accuracy
ST2
R
ST1
Aug1
ST0
Aug0
Local Ephemeral Storage
Performance tier
Capacity tier
Interference
t1
t2
t
w(|Aug_{εm}^{L(εm)}|, εm, p)
2 Interference measurement and estimation
Target container
Interfering containers
3 Runtime weight control and error bounded recomposition
High interference at t1
Stay within the performance tier
Offer two levels of accuracy
Accuracy level #0: R
Accuracy level #1: R ⊕ Aug1
Medium/Low interference at t2
Can access the capacity tier
Runtime weight control
Additionally offer a higher accuracy
Accuracy level #2: R ⊕ Aug1 ⊕ Aug0

**Fig. 3:** An illustration of Tango. Herein we assume the interference is over the shared $ST^0$ and $ST^1$. The data analytics can be performed in three accuracy levels. If there is interference experienced, the data analytics can adapt and lower the accuracy, so that there is less or no data retrieved from the capacity tier. The operator $\oplus$ denotes the recomposition.

subsequent steps (step ❷). Based upon the estimated storage bandwidth, data analytics can adjust the augmentation and the weight of blkio on-the-fly (step ❸) at a future timestep $t_s$. As an example in Fig. 3, if a low accuracy is requested by the user, data analytics retrieves the base representation $R = \Omega^2$ only from $ST^2$. If a medium accuracy is rather desired, the base reduced representation will be further augmented by fetching $Aug^1$ from $ST^1$ and recomposing $\Omega^1$. If there is a low interference, the data analytics may further access the capacity tier by fetching $Aug^0$ to elevate the accuracy to the highest, i.e., $\Omega^0$. More details are described as follows.

## B. Error-bounded Refactorization

*1) Error Metric:* We enforce error control during the dynamic augmentation to limit information loss and reduce I/O interference. To achieve this, we use NRMSE and PSNR to characterize and control the error between reconstructed data and full data. For the former, we first calculate root mean square error (RMSE), which is the square root of the average of squared differences between the predicted and actual value, further normalized by the data range of the quantity. Meanwhile, PSNR is the ratio between the maximum value of a signal and the power of distorting noise that affects the quality of its representation. The calculations of these two error metrics are shown below.

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2}}{x_{max} - x_{min}}$$

$$PSNR = 10\log_{10}\frac{x_{max}^2}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2}$$

*2) Hierarchical Decomposition with Error Control:* To allow the augmentations to be incrementally retrieved, data needs to be decomposed into hierarchical levels. In this work, we treat the analysis output from a simulation as a tensor (or a uniform grid) and the decomposition consists of the following iterative steps until the reduced representation $R$ is satisfactory (e.g., in terms of either size or accuracy), as shown in Fig 4.

**TABLE III:** A list of notations.

| Notation | Description |
|---|---|
| $abplot(\cdot)$ | Augmentation-bandwidth plot. |
| $Aug^l$ | The augmentation that elevates the representation from level $l+1$ to $l$. |
| $Aug^l_{(x,y)}$ | The data point at (x, y) in the augmentation that elevates the representation from level $l+1$ to $l$. |
| $Aug^l_{\epsilon_i,(x,y)}$ | The set of data points in the augmentation that elevates the accuracy from $\epsilon_{i-1}$ to $\epsilon_i$. |
| $b$ | Number of error bounds. |
| $BW_{low}$, $BW_{high}$ | Lower/higher bandwidth threshold in the augmentation-bandwidth plot. If the predicted bandwidth is lower/higher than this, storage is deemed to be highly/lightly congested. |
| $BW_i$, $\tilde{BW_i}$ | Measured/estimated bandwidth at a step $i$. |
| $d^l$ | Decimation ratio of level $l$. |
| $\epsilon_i$ | The $i$-th error bounds. |
| $FC_i$ | The amplitude of the $i$-th frequency component. |
| $\tilde{FC_i}$ | The amplitude of the $i$-th frequency component after thresholding. |
| $L$ | The maximum number of levels. |
| $\mathcal{L}(\epsilon_i)$ | The level that achieves error bound $\epsilon_i$. |
| $\Omega^l$ | The $l$-th level data representation. |
| $\Omega^l_{(x,y)}$ | The data point at (x, y) in $l$-th level data representation. |
| $p$ | Priority of data analytics. |
| $prolongate(\cdot)$ | A linear function that provides interpolation from level $l+1$ back to $l$. |
| $ST^l$ | The $l$-th storage tier. |
| $thresh$ | Threshold of frequency amplitude. |
| $R$ | Reduced data representation. |
| $restrict(\cdot)$ | A linear function that performs restriction from level $l$ to $l+1$. |
| $t_s$ | The $s$-th timestep. |
| $w(\cdot)$ | A weight function that maps the cardinality of augmentation, the associated accuracy, and priority to container weight. |

**Step 1: restriction.** We restrict the tensor from level $l$ to $l+1$ by retaining every $d^l$-th data point along each dimension, i.e., $\Omega^{l+1} = restrict(\Omega^l)$. Let a data point at $(x, y)$ in $\Omega^l$ be $\Omega^l_{(x,y)}$. Fig. 4 shows an example of restricting a 2D tensor with $d^l = 2$. After the restriction, the top left element is as follows, $\Omega^{l+1}_{(0,0)} = \Omega^l_{(0,0)}$, $\Omega^{l+1}_{(0,1)} = \Omega^l_{(0,2)}$, $\Omega^{l+1}_{(1,0)} = \Omega^l_{(2,0)}$, $\Omega^{l+1}_{(1,1)} = \Omega^l_{(2,2)}$.

**Step 2: prolongation and subtraction.** To compute the

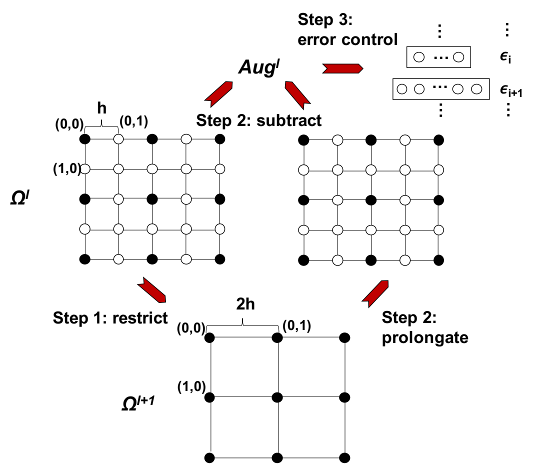


**Fig. 4:** Decomposition. Here we decompose $\Omega^l$ to $\Omega^{l+1}$ and $Aug^l$.

augmentation $Aug^l$, we prolongate $\Omega^{l+1}$ back to level $l$ and subtract $\Omega^l$ from the prolongation, i.e., $Aug^l = prolongate(\Omega^{l+1}) - \Omega^l$. Let a data point at $(x, y)$ in $Aug^l$ be denoted as $Aug^l_{(x,y)}$. Those data points that are on both levels do not need to be explicitly stored since they are zeros after subtraction, i.e., $Aug^l_{(0,0)} = Aug^l_{(0,2)} = Aug^l_{(2,0)} = Aug^l_{(2,2)} = 0$. For the rest of the data points, under a linear interpolation, $Aug^l_{(0,1)} = \frac{1}{2}(\Omega^{l+1}_{(0,0)} + \Omega^{l+1}_{(0,1)}) - \Omega^l_{(0,1)}$, $Aug^l_{(1,0)} = \frac{1}{2}(\Omega^{l+1}_{(0,0)} + \Omega^{l+1}_{(1,0)}) - \Omega^l_{(1,0)}$, $Aug^l_{(1,1)} = \frac{1}{4}(\Omega^{l+1}_{(0,0)} + \Omega^{l+1}_{(0,1)} + \Omega^{l+1}_{(1,0)} + \Omega^{l+1}_{(1,1)}) - \Omega^l_{(1,1)}$. Other data points can be calculated in a similar fashion and are not detailed further.

**Step 3: error control.** Let the range of error bounds be $\{\epsilon_i, 0 \leq i < b\}$, where $b$ is the number of error bounds and $\epsilon_i$ represents an accuracy lower than $\epsilon_{i+1}$. After $Aug^l$ is constructed, all data points in $Aug^l$ will be sorted by the absolute value $|Aug^l_{(x,y)}|$ in descending order–clearly, a larger absolute value indicates a greater impact on the overall error, therefore needs to be retrieved earlier once the degree of augmentation is determined. Further, the subset of data points that need to be retrieved to reach a given error bound $\epsilon_i$, denoted as $\bigcup\limits_{(x,y)} Aug^l_{\epsilon_i,(x,y)}$, will be shuffled together and properly tagged, such that when $\epsilon_i$ is requested later by data analytics, $\bigcup\limits_{(x,y)} Aug^l_{\epsilon_i,(x,y)}$ can be rapidly identified and retrieved. If an additional augmentation is needed to elevate the accuracy from $\epsilon_i$ to $\epsilon_{i+1}$, $\bigcup\limits_{(x,y)} Aug^l_{\epsilon_{i+1},(x,y)}$ will be further retrieved and applied to the reduced representation. Overall, this shuffling is done not only for the purpose of rapidly achieving a user-prescribed error bound, but also to ensure the access of augmentations is mostly contiguous on disk to achieve a higher I/O performance.

For a 1D dataset with $n$ data points, the complexity of decomposition is $O(nlog(n))$, since the algorithm will be iterated for $log(n)$ times and each iteration has a complexity of $O(n)$.

**Algorithm 1** Error-bounded cross-layer recomposition.

```
Require: Base representation $R$, a set of augmentations $\bigcup\limits_{0\leq l<L-1} Aug^l$, a
    prescribed error bound $\epsilon_i$, priority $p$, a target timestep $t_s$
Ensure: A recomposed representation with an accuracy satisfying $\epsilon_i$
 1: Retrieve the base representation $R$ from $ST^{L-1}$
 2: Collect the performance of data analytics for $n$ consecutive steps $\{BW_i\}$
 3: Perform DFT over $\{BW_i\}$ to convert the measurements to the frequency
    domain, i.e., $\{FC_i\} \leftarrow DFT(\{BW_i\})$
 4: Perform thresholding over $\{FC_i\}$ and obtain $\{\tilde{FC_i}\}$. In particular, if
    $FC_i < thresh$, $\tilde{FC_i} \leftarrow 0$
 5: Perform inverse DFT, i.e., $\{\tilde{BW_i}\} \leftarrow IDFT(\{\tilde{FC_i}\})$, and obtain $\tilde{BW_s}$
    for the target timestep $t_s$
 6: Compute the augmentation degree based upon the augmentation-
    bandwidth plot $abplot(\tilde{BW_s})$ and in turn the accuracy level $\epsilon_j$ that can
    be achieved
 7: $k \leftarrow max(i, j)$
 8: $m \leftarrow 1$
 9: while $m \leq k$ do
10:     Compute the weight function $w(|Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}|, \epsilon_m, p)$ and apply it to
        the blkio
11:     Retrieve $Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}$ from $ST^{\mathcal{L}(\epsilon_m)}$
12:     $m \leftarrow m + 1$
13: end while
14: $m \leftarrow 0$, $prev \leftarrow 0$, $cur \leftarrow 0$, $r \leftarrow R$
15: while $m \leq k$ do
16:     $cur \leftarrow \mathcal{L}(\epsilon_m)$
17:     if $prev > cur$ then
18:         $r \leftarrow prolongate(r) + Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}$
19:     else
20:         $r \leftarrow r + Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}$
21:     end if
22:     $prev \leftarrow cur$, $m \leftarrow m + 1$
23: end while

24: return $r$
```

### C. Cross-layer Interference Mitigation

*1) Dynamic Recomposition with Error Control:* During data analysis, we retrieve the base representation $R$ possibly alongside a set of augmentations $\subset \bigcup\limits_{0\leq l<L-1} Aug^l$, where $L$ denotes the maximum number of levels. If the resulting accuracy of $R$, termed as base accuracy $\epsilon_0$, is lower than the user-prescribed error bound $\epsilon_i, i > 0$, data analytics must further augment the accuracy of data by fetching adequate augmentations, denoted as $\bigcup\limits_{m=1}^{i} Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}$, from lower storage tiers regardless of the intensity of interference, where $\mathcal{L}(\epsilon_m)$ denotes the level that achieves the error bound $\epsilon_m$. Based upon the estimated interference, the data analytics may further augment the accuracy on-the-fly by fetching more augmentations. For example, it may augment the accuracy to $\epsilon_j, j > i$ by retrieving $\bigcup\limits_{m=i+1}^{j} Aug^{\mathcal{L}(\epsilon_m)}_{\epsilon_m}$, if the intensity of interference is estimated to be insignificant. Algorithm 1 shows the error-bounded recomposition algorithm. In particular, the recomposition consists of the following steps:

**Step 1: interference measurement and estimation.** The goal of interference estimation is to take advantage of the periodic nature of HPC workloads and estimate the interference on storage for a future timestep $t_s$, so that an appropriate amount of augmentation can be determined–the higher the interference is, the lower the augmentation will be. While the HPC application patterns are periodic and can be described as $\sum_i I_i(C_i^x W_i)^* F_i$, the reality is that the storage workload is complex and dynamic since applications come and go and involve some degree of randomness, such as those incurred by code compilation and ad-hoc user commands. In this work, we adopt a signal processing based technique that will be periodically performed (e.g., in every 45 steps) to estimate the storage interference. In particular, we collect the performance of data analytics in early steps and use DFT to identify the frequency components with an amplitude greater than $thresh$, so that those interferences that are relatively low in impact (i.e., non-recurrent and random noise) can be discarded. Next, we perform an inverse DFT (IDFT) to convert the signal back to the time domain so that we can obtain the amplitude of interference at $t_s$ and in turn the available bandwidth $\tilde{BW_s}$. The estimation will be done periodically so that when the interference pattern changes, the estimation can be re-adjusted. We note that the complexity of DFT/IDFT is $O(n \cdot log(n))$ and therefore the overhead of estimation is low.

**Step 2: prolongation and addition.** After the interference intensity at $t_s$ is estimated, it will be further proportionally mapped to the degree of augmentation (i.e., the percentage of augmentations) to be performed–when there is no interference, a full augmentation is done, while when the interference exhausts all storage bandwidth, no augmentation is done (if no error control). Otherwise, the degree of augmentation is a linear function of the available bandwidth, i.e., $abplot(\tilde{BW_s}) = k_1 \cdot \tilde{BW_s} + b_1 \in [0, 1]$, where $k_1$ and $b_1$ are coefficients. For a given storage system, if the estimated bandwidth is no less than $BW_{max}$, the storage system is considered to be lightly loaded and therefore a full augmentation will be done, i.e., if $\tilde{BW_s} \geq BW_{max}$, $abplot(\tilde{BW_s}) = 1$. On the other hand, if the estimated bandwidth is no greater than $BW_{min}$, the storage system is considered to be heavily loaded, and therefore no additional augmentation will be done (beyond those for satisfying the prescribed error bound) to avoid over-loading the system [35], i.e., if $\tilde{BW_s} \leq BW_{min}$, $abplot(\tilde{BW_s}) = 0$. Based upon these two conditions, we can compute $k_1$ and $b_1$ and determine $abplot(\tilde{BW_s})$ when $BW_{min} < \tilde{BW_s} < BW_{min}$. Once the amount of augmentations and in turn the accuracy, say $\epsilon_i$, to be retrieved is known, the following recomposition will be performed by prolongating $\Omega^{\mathcal{L}(\epsilon_1)+1}$ to level $\mathcal{L}(\epsilon_1)$ and applying the augmentation, i.e., $\Omega^{\mathcal{L}(\epsilon_1)} = prolongate(\Omega^{\mathcal{L}(\epsilon_1)+1}) + Aug_{\epsilon_1}^{\mathcal{L}(\epsilon_1)}$. Similarly, if $\mathcal{L}(\epsilon_2) = \mathcal{L}(\epsilon_1)$, $\Omega^{\mathcal{L}(\epsilon_2)} = prolongate(R) + Aug_{\epsilon_1}^{\mathcal{L}(\epsilon_1)} + Aug_{\epsilon_2}^{\mathcal{L}(\epsilon_2)}$. Otherwise, $\mathcal{L}(\epsilon_2) = \mathcal{L}(\epsilon_1) - 1$, and $\Omega^{\mathcal{L}(\epsilon_2)} = prolongate(\Omega^{\mathcal{L}(\epsilon_1)}) + Aug_{\epsilon_2}^{\mathcal{L}(\epsilon_2)}$. These operations will be repeated until we reach $\epsilon_i$.

**Step 3: blkio weight adjustment.** At the storage layer, the blkio weight of the associated container will be adjusted at $t_s$ in order to cooperate with the augmentation constructed. Such a design is motivated by the fact that a larger augmentation requires more storage allocation than a smaller augmentation, and without adjusting blkio weight, all applications would share equal bandwidth, which can lead to sub-optimal outcomes (as shown in Fig. 9 in Section IV). We carefully note that the weight adjustment requires neither the administrator access nor restarting the container to become effective, and data analytics can adjust its weight on-the-fly at each iteration.

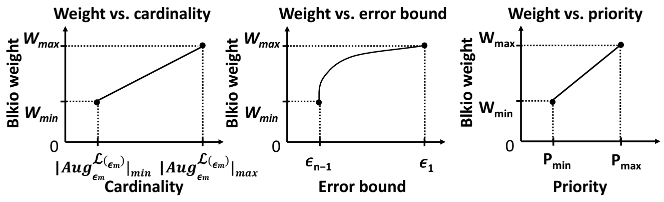


**Fig. 5:** A schematic of weight function.

In this work, the blkio weight is adjusted at runtime according to a weight function $w(|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}|, \epsilon_m, p)$ that takes the cardinality of an augmentation $|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}|$, the associated accuracy level $\epsilon_m$, and the priority $p$ of an application. The central idea behind this design is three-fold: 1) the storage layer must cooperate with the application layer by allocating commensurate resources for augmentation. Intuitively, the weight of a container should be proportional to the cardinality of the total augmentations so that a large augmentation will be allocated more bandwidth, as opposed to an equally divided share with the interference; 2) the latency to retrieve a lower accuracy $\epsilon_m$ is more important than the latency to retrieve an augmentation to elevate the accuracy to $\epsilon_{m+1}$. The reason is that the low-accuracy data generally contains critical information (Fig. 2), and for many interactive data analytics, the data needs to be processed rapidly to ensure timely actions can be made; 3) on the other hand, the blkio weight also needs to be adjusted on a per-application basis. For interactive data analytics, the weight should be set higher than those offline data analytics to ensure a faster response time. Therefore, we further introduce the notation of priority $p$ to provide differential services to applications. In this work, the weight function is designed as $w(|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}|, \epsilon_m, p) = k_2 \cdot \frac{|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}| \cdot p}{|lg(\epsilon_m)|} + b_2$ for NRMSE and $w(|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}|, \epsilon_m, p) = k_2 \cdot \frac{|Aug_{\epsilon_m}^{\mathcal{L}(\epsilon_m)}| \cdot p}{|\epsilon_m|} + b_2$ for PSNR, where $k_2$ and $b_2$ are coefficients. In particular, coefficient $k_2$ determines how aggressively that data analytics can request storage bandwidth. In general, $k_2$ and $b_2$ can be determined as follows. We determine the lowest accuracy, the highest priority, and the largest cardinality of data that can be possibly achieved, which corresponds to the scenario of the maximum weight (e.g., 1000 in Docker container). Similarly, we determine the highest accuracy, the lowest priority, and the lowest cardinality, which corresponds to the scenario of the minimum weight (e.g., 100 in Docker container). Given these two conditions, we can obtain $k_2$ and $b_2$. In Fig. 5, we illustrate the relationship between the weight and cardinality, accuracy, and priority, respectively. The recomposition is the inverse of decomposition, and therefore the complexity of the algorithm is $O(nlog(n))$, too. As such, this paper only focuses on the performance of recomposition in the performance evaluation.

## IV. Evaluation

### A. Testbed

We use the Chameleon experimental facility [27] at the University of Chicago and Texas Advanced Computing Center to evaluate Tango. A physical compute node in our testbed (shown in Fig. 6) has an Intel Xeon 2.3GHz processor with 10 cores, an Intel 400GB SATA SSD, and a Seagate 2TB 7200 RPM SAS HDD. The file system used for SSD and HDD is Ext4. Based upon the hardware availability, we build a two-tier local storage system with the fast tier being SSD and the slow tier being HDD. Since the algorithms of Tango are embarrassingly parallel, we obtained most results on a single node, while for the scalability test, we used 4 compute nodes.

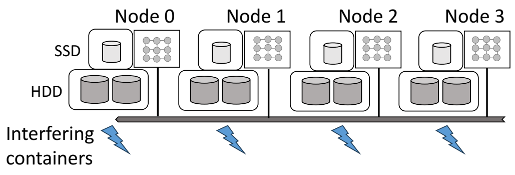


**Fig. 6:** Testbed.

The applications were executed through the Docker container, with each container hosting one executable (either data analytics or noise). By default, with each node, we configure one container to run the data analytics, alongside six interfering containers (see Table IV) that inject periodic I/O interference to HDD, mimicking those checkpointing activities from simulations. Unless otherwise noted, the decimation ratio used to construct the reduced representation is 16, and the default blkio weight of each container is 100. The estimation is performed for every 30 timesteps and the period of data analytics is 60 seconds. For the DFT-based estimation method, we extract the noises whose frequency components are higher than 50 % of the maximum amplitude. For the augmentation-bandwidth plot, $BW_{low}$ and $BW_{high}$ are set to 30 MB/s and 120 MB/s, respectively. For the priority parameter in the weight function, we test three settings, i.e., 1 (*low*), 5 (*medium*), and 10 (*high*), for data analytics to understand the impact of priority to the I/O performance.

**TABLE IV:** Noise injected to HDD.

| Noise | Period | Checkpoint Size |
|---|---|---|
| Interfering container #1 | 200 secs | 768 MB |
| Interfering container #2 | 225 secs | 512 MB |
| Interfering container #3 | 360 secs | 512 MB |
| Interfering container #4 | 180 secs | 1024 MB |
| Interfering container #5 | 150 secs | 1024 MB |
| Interfering container #6 | 120 secs | 1024 MB |

We test three applications, XGC, GenASiS, and CFD. For XGC, we conduct *blob detection* [36], [37] for the data produced by the XGC simulation. These blobs represent physical regions with high electrostatic potentials that are often of interest to fusion scientists. The data analytics examines the characteristics of the *dpot* dataset and measures how much the electric potential deviates from the background. The dataset consists of 89,857,269 triangles in the mesh. GenASiS [38] is a multi-physics code developed for the simulation of astrophysical systems involving nuclear matter. The data analytics is a simple 2D rendering of the velocity magnitude of core-collapse. The mesh consists of 94,806,450 triangles. CFD studies and analyzes the interaction of liquids with surfaces under certain boundary conditions. This data analytics examines the pressure near the front of a plane and its mesh consists of 61,529,058 triangles.

For XGC, we measure the error of the analysis outcome with regard to the characteristics of the blobs detected, e.g., the number of blobs, average blob diameter, etc. For GenASiS, we perform the 2D rendering of the velocity of core-collapse and measure the structural similarity index (SSIM) [39] and Dice's coefficient [40] between the reduced and original representations. For CFD, we measure the error regarding the total area with a high pressure and the total force on the high-pressure area.

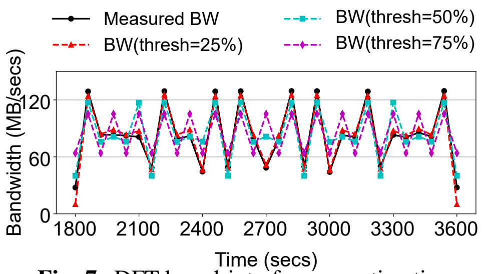


**Fig. 7:** DFT-based interference estimation.

### B. Cross-layer vs. Single-layer

Fig. 7 demonstrates the effectiveness of the DFT-based estimation with six interfering containers, with $thresh$ of 25%, 50%, and 75%, respectively. Herein we use the performance measured during the first 1800 secs to obtain $\tilde{BW}$ for the subsequent 1800 secs (1800 - 3600 sec). Overall, the estimated bandwidth is highly accurate, despite that the low amplitude frequency components are discarded. As $thresh$ becomes larger, more components are discarded, therefore the estimation deviates more from the real bandwidth. Overall, the accurate estimation provides the basis for our dynamic augmentation.

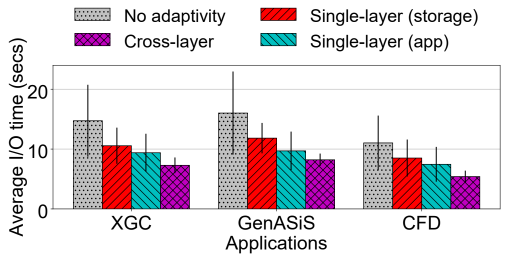


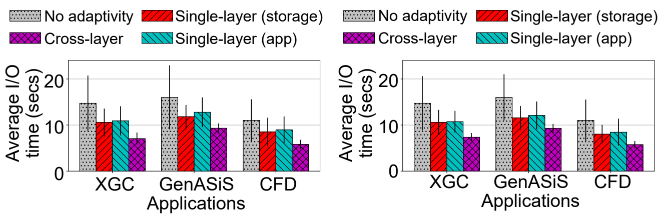
**Fig. 8:** Cross-layer vs. single-layer with no error control. For the single-layer approach with storage adaptivity, we perform a full augmentation and the blkio weight is set proportionally according to the augmentation size. For the single-layer approach with application adaptivity, we perform the dynamic augmentation based upon the estimated interference. For comparison, no adaptivity is also measured where blkio weight is set to 100 with a full augmentation.

We next assess the effectiveness of the cross-layer approach. In particular, we compare the cross-layer approach with the single-layer approach where 1) only the blkio weight is adjusted at the storage layer according to the size of data to be retrieved, without performing the dynamic augmentation; 2) only the dynamic augmentation is done at the application layer without support from the storage layer (i.e., [3], [2]). For comparison, we also test the baseline where the full augmentation is always done without weight adjustment, i.e., no storage or application layer adaptivity at all. Fig. 8 shows the average I/O time and variation (measured by the error bar) of XGC, GenASiS, and CFD when no error control is enforced. As such, the augmentation retrieved is completely based upon the estimated storage load. It is clear that the baseline case, which represents how applications access data from storage conventionally, is static in nature and therefore yields the highest I/O time and variation. With either the storage or application adaptivity introduced to data retrieval, the overall I/O performance improves. On the other hand, the application-layer is shown to be more effective in mitigating the interference than the storage-layer approach. The rationale behind this is that adjusting the weight at the storage layer is only effective when the storage bandwidth is under-utilized. Once the storage system becomes highly loaded, the weight adjustment only re-distributes the bandwidth among applications without fundamentally alleviating the I/O congestion. Overall, the cross-layer approach yields the best performance by incorporating both application- and storage-layer control.

## C. Error-bounded Interference Mitigation

We further evaluate the performance of interference mitigation with error control. Fig. 9 shows the average retrieval time with $\epsilon$ of 0.01 for NRMSE and 30 for PSNR, respectively. Intuitively, the error control enforces the minimal amount of augmentations that must be retrieved, and as a result, the performance of cross-layer and single-layer with application adaptivity may degrade as compared to the performance without error control (Fig. 8).

Fig. 10 evaluates the data quality by measuring the relative error of the analysis outcome at a loose error bound of 0.1 for NRMSE. Other tighter error bounds (e.g., 0.01 etc.) result in smaller deviations than the results shown here. We compare the cross-layer to the single-layer with application adaptivity. Note that the single-layer with storage adaptivity does not lose accuracy and therefore is not shown here. The case of no augmentation represents the scenario where data analytics does not perform augmentation at all and only retrieves the base representation from SSD with high bandwidth, thus



**(a)** $\epsilon = 0.01$ (NRMSE). **(b)** $\epsilon = 30$ (PSNR).

**Fig. 9:** Interference mitigation with error control. Note that *NRMSE* and *PSNR* are used here to control the error. The scenarios of no adaptivity and singe-layer with storage adaptivity retrieve a full augmentation and therefore do not need to be error-controlled.

serving as the worst possible scenario with regard to the data quality. By introducing the error control in the cross-layer, more augmentations may be retrieved by the cross-layer, thus resulting in a higher data quality than the single-layer.

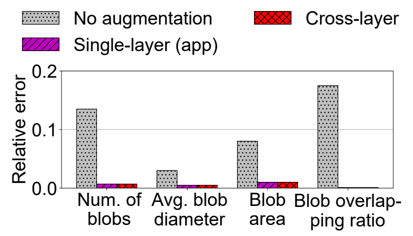
Fig. 11 further measures the percentage of the degree of freedom retrieved across error bounds. It is observed that retrieving less than 30% of data can maintain $\epsilon$ of 0.00001 for NRMSE and 80 for PSNR, which demonstrates the feasibility of Tango in trading accuracy for performance.

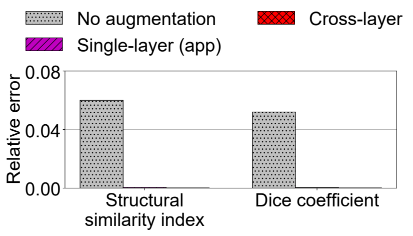
Fig. 12 evaluates the impact of noise intensity by varying the number of interfering containers over HDDs. It is observed that the cross-layer is rather insensitive to the increase of noises, while the mean and variance of the single-layer with storage adaptivity degrades substantially with the intensity of interference. As the number of interfering contains increases, the performance gain of cross-layer versus single-layer becomes more pronounced.

## D. Weight Adjustment

We next evaluate the impact of weight assignment in blkio on data analytics. Fig. 13 illustrates the I/O performance of elevating the accuracy from base accuracy (achieved by $R$) to $\epsilon_1 = 0.01$ for a high priority ($p = 10$) data analytics. In particular, we study the scenarios where the weight assignment is affected by 1) the cardinality of augmentation only; 2) both cardinality and priority; and 3) cardinality, priority, and accuracy. The performance of the single-layer approach is also shown here as the baseline. The average I/O time measured here represents the latency for the data analytics to retrieve a low accuracy augmentation, a metric of high importance for online data analytics. It is found that as we progressively incorporate these parameters into the weight function, the latency is gradually improved due to the higher allocation of storage resources. Fig. 14 further shows the cross-layer performance under a range of priorities and error bounds. In particular, Fig. 14a shows the impact of priority at a fixed $\epsilon$ of 0.01, while Fig. 14b shows the impact of the error bound with a fixed $p$ of 10. While the overall trends confirm the effectiveness of our design, we want to point out that a 2X increase in priority from 5 to 10 does not mean the doubling of storage resources. For example, assume there are two containers with equal weights of 100 competing for storage bandwidth totaling 200 MB/s. When doubling the weight from 100 to 200 for one container, it will only increase its bandwidth allocation from 100 MB/s to 133 MB/s. Fig. 15 illustrates the weight assignment for XGC from 1800 to 1950 secs. It can be seen that as the accuracy increases from 1e-2 to 1e-4, the weight is gradually lowered mostly due to our design that favors a low accuracy.

Fig. 16 shows the scaling performance of Tango using weak scaling. The recomposition operation in Tango can be done locally without communication. Given the embarrassingly parallel nature of Tango, the average I/O stays the same when the number of compute nodes increases from 1 to 4.


**(a)** XGC.


**(b)** GenASiS.

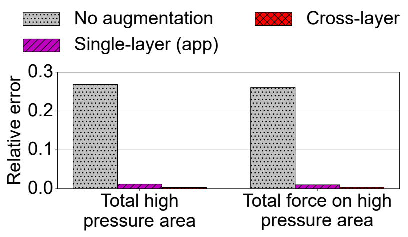

**(c)** CFD.

**Fig. 10:** Data quality evaluated through the data analysis results. Here the priority is 10, and the decimation ratio is 8192. The error bound $\epsilon$ is set to 0.1 (NRMSE).

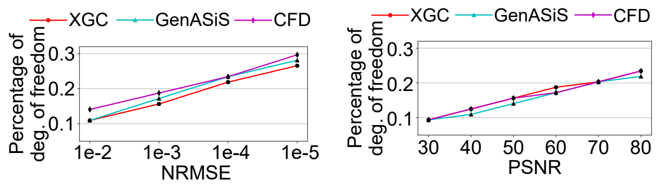


**Fig. 11:** Percentage of the degree of freedom vs. error bound.

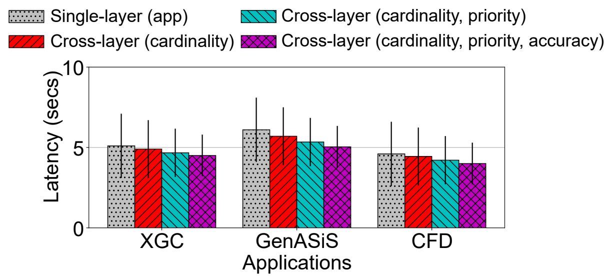


**Fig. 13:** The latency to retrieve the augmentation that elevates the accuracy to 0.01 (NRMSE). Note that the latency for single-layer with storage adaptivity is identical to cross-layer with cardinality, and therefore is not shown.

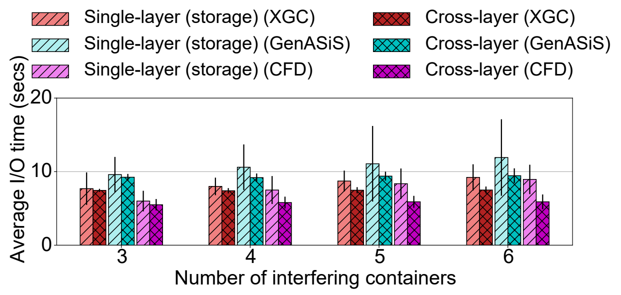
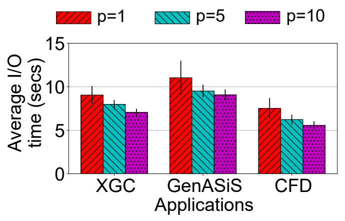


**Fig. 12:** Performance of cross-layer vs. the number of noises. The priority is 10 and the target NRMSE is 0.01. For the first three noises, we inject containers #1, #2, and #3. We then incrementally inject containers #4, #5, and #6.


**(a)** $\epsilon = 0.01$ (NRMSE).

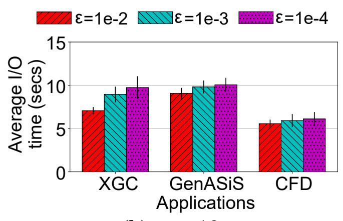

**(b)** $p = 10$.

**Fig. 14:** Impact of priority and error bound.

## V. Related Work

In-memory computing is a new paradigm proposed to control the I/O overhead of HPC simulations. Based on how the simulation and data analytics are coupled, there are two major categories of in-memory computing: in situ [41], [42], [43], [7] and in transit [44], [45], [46], [47], [48]. For the former, the compute resource of data analytics is allocated to the same node where the simulation is. For the latter, the data will move from the simulation memory to a dedicated staging area before data analysis, which benefits simulation by allowing it to run asynchronously with the data analysis.

For large-scale systems, to improve the simplicity, efficiency, and effectiveness of data analysis, new data management frameworks, such as ADIOS [10] and Mochi [49], have been proposed to provide multiple methods that can be easily integrated into the applications for scientists to use. In the meantime, to enhance the I/O performance, new methods were proposed for data placement and organization [50], [9], [51]. However, these prior works were limited to improving either peak or average I/O performance, ignoring the importance of consistency of performance. To address the issue of the performance on large HPC systems, prior work [52], [19], [18] balances the I/O requests between busy and idle devices to mitigate the impact of storage interference. Prior work [53] further digs into the root cause of the interference in storage systems. In general, these works mostly focus on middleware-level solutions, without exploring the application level.

Data reduction is another well-known research area to manage the exponential growth of data. Data reduction can be either lossless compression or lossy compression. Lossless compression, FPC [12], FPZIP [54], GZIP [55], has strong requirements that the data should be exactly the same as it was before after compression and decompression. Lossless compression is always used on accuracy-sensitive scenarios, such as restart and checkpoint. But it can only reach low compression ratios, which restricts availability, especially in the case that the data from simulation can easily grow exponentially. In such circumstances, lossy compression, SZ [13], ZFP [14], ISABELA [56] can meet our demand by providing higher compression ratios with low overhead by trading accuracy for performance. In this paper, the reduced representations are generated by data reduction which is one of the forms of lossy compression by retaining some selected data points. However we should pay more attention that lossy compression can lead to information loss, thus this work performs additional

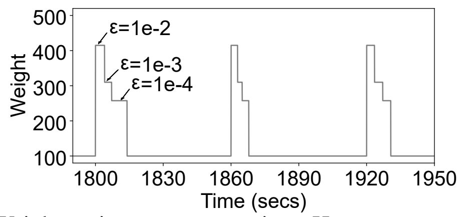


**Fig. 15:** Weight assignment across time. Here we measure how the weight is adjusted during the data analysis for XGC. The priority is set to 10 and the target NRMSE is 0.01 for error control.

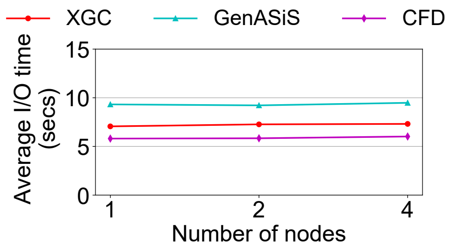


**Fig. 16:** The scalability of Tango. The priority is set to 10 and the error bound is set to 0.01 (NRMSE).

augmentation to address this issue.

## VI. Conclusion

This paper aims to address the issue of I/O interference for data analytics on local ephemeral storage. In particular, we take advantage of proportional resource control in the emerging scenarios of containerization, and propose a coordinated cross-layer approach that reacts to storage interference from both storage and application layers, while maintaining a prescribed error bound to limit the information loss. We evaluate three data analytics, XGC, GenASiS, and CFD, and quantitatively demonstrate that the I/O performance can be vastly improved as opposed to the single-layer approach, while maintaining acceptable outcomes of data analysis.

## VII. Acknowledgments

This research was supported by the DOE SIRIUS-2 ASCR research project and the National Science Foundation under Grant OAC-2311757, CCF-2134202, and OAC-2144403.

## References

[1] S. Jin, D. Tao, H. Tang, S. Di, S. Byna, Z. Lukic, and F. Cappello, "Accelerating parallel write via deeply integrating predictive lossy compression with hdf5," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '22. IEEE Press, 2022.

[2] X. Liang, Q. Gong, J. Chen, B. Whitney, L. Wan, Q. Liu, D. Pugmire, R. Archibald, N. Podhorszki, and S. Klasky, "Error-controlled, progressive, and adaptable retrieval of scientific data with multilevel decomposition," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476179

[3] Z. Qiao, Q. Liu, N. Podhorszki, S. Klasky, and J. Chen, "Taming i/o variation on qos-less hpc storage: What can applications do?" in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–13.

[4] T. Lu and et al., "Canopus: A paradigm shift towards elastic extreme-scale data analytics on hpc storage," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017, pp. 58–69.

[5] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci *et al.*, "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. IEEE, 2012, pp. 1–9.

[6] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf, "Predata–preparatory data analytics on peta-scale machines," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–12.

[7] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling in-situ execution of coupled scientific workflow on multi-core platform," in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. IEEE, 2012, pp. 1352–1363.

[8] M. Gamell, I. Rodero, M. Parashar, J. C. Bennett, H. Kolla, J. Chen, P.-T. Bremer, A. G. Landge, A. Gyulassy, P. McCormick *et al.*, "Exploring power behaviors and trade-offs of in-situ data analytics," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–12.

[9] W. Liao and A. Choudhary, "Dynamically adapting file domain partitioning methods for collective i/o based on underlying parallel file system locking protocols," in *SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE, 2008, pp. 1–12.

[10] Q. Liu and et al., "Hello adios: The challenges and lessons of developing leadership class i/o frameworks," *Concurr. Comput. : Pract. Exper.*, vol. 26, no. 7, pp. 1453–1473, May 2014.

[11] T. Wang, K. Mohror, A. Moody, K. Sato, and W. Yu, "An ephemeral burst-buffer file system for scientific applications," in *SC'16*, 2016.

[12] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2009.

[13] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, 2016, pp. 730–739.

[14] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[15] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[16] T. Patel, S. Byna, G. K. Lockwood, and D. Tiwari, "Revisiting i/o behavior in large-scale storage systems: the expected and the unexpected," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356183

[17] M. Dorier, G. Antoniu, R. Ross, D. Kimpe, and S. Ibrahim, "Calciom: Mitigating i/o interference in hpc systems through cross-application coordination," in *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, ser. IPDPS'14. USA: IEEE Computer Society, 2014, pp. 155–164. [Online]. Available: https://doi.org/10.1109/IPDPS.2014.27

[18] D. Huang, Q. Liu, J. Choi, N. Podhorszki, S. Klasky, J. Logan, G. Ostrouchov, X. He, and M. Wolf, "Can i/o variability be reduced on qos-less hpc storage systems?" *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 631–645, May 2019.

[19] Q. Liu, N. Podhorszki, J. Logan, and S. Klasky, "Runtime i/o re-routing + throttling on hpc storage," in *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, ser. HotStorage'13. USA: USENIX Association, 2013, p. 14.

[20] "Compute - amazon ec2 instance types - aws." [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[21] "Intel xeon phi processor 7210." [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/94033/intel-xeon-phi-processor-7210-16gb-1-30-ghz-64-core/specifications.html

[22] "Sharing Consumable Resources in SLURM." [Online]. Available: https://slurm.schedmd.com/cons_tres_share.html

[23] "Center for high performance computing at university of utah." [Online]. Available: https://www.chpc.utah.edu/documentation/software/node-sharing.php

[24] "High performance computing (hpc) at university of maryland (umd)." [Online]. Available: https://hpcc.umd.edu/hpcc/help/jobs.html

[25] "Advanced research computing at hopkins." [Online]. Available: https://https://www.arch.jhu.edu/support/slurm-queueing-system/

[26] D. Dai, Y. Chen, D. Kimpe, and R. Ross, "Two-choice randomized dynamic i/o scheduler for object storage systems," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 635–646.

[27] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, *Chameleon: a Scalable Production Testbed for Computer Science Research*, ser. In: Jeffrey Vetter (eds) Contemporary High Performance Computing: From Petascale toward Exascale. Boca Raton, FL: Chapman & Hall/CRC Computational Science, CRC Press, 2019.

[28] Y. Qian, X. Li, S. Ihara, L. Zeng, J. Kaiser, T. Süß, and A. Brinkmann, "A configurable rule based classful token bucket filter network request scheduler for the lustre file system," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126932

[29] "Spectrum scale," 2024, https://www.ibm.com/docs/en/gpfs/4.1.0.4, Accessed: 3-25-2024.

[30] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, ser. OSDI '06. USA: USENIX Association, 2006, pp. 307–320.

[31] A. Gulati, A. Merchant, and P. J. Varman, "Mclock: Handling throughput variability for hypervisor io scheduling," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. USA: USENIX Association, 2010, pp. 437–450.

[32] "Orange file system 2.9.7," 2021, http://www.orangefs.org, Accessed: 3-25-2024.

[33] L. T. Yang, X. Ma, and F. Mueller, "Cross-platform performance prediction of parallel applications using partial execution," in *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, Nov 2005, pp. 40–40.

[34] Y. Jin, X. Ma, M. Liu, Q. Liu, J. Logan, N. Podhorszki, J. Y. Choi, and S. Klasky, "Combining phase identification and statistic modeling for automated parallel benchmark generation," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS'15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 309–320. [Online]. Available: https://doi.org/10.1145/2745844.2745876

[35] G. Somasundaram and A. C. Shrivastava, "Information storage and management : storing, managing, and protecting digital information in classic, virtualized, and cloud environments." John Wiley & Sons, 2012, pp. 39–40.

[36] D. A. D'Ippolito, J. R. Myra, and S. J. Zweben, "Convective transport by intermittent blob-filaments: Comparison of theory and experiment," *Physics of Plasmas*, vol. 18, no. 6, p. 060501, Jun. 2011.

[37] S. Ku, R. M. Churchill, C. S. Chang, R. Hager, E. S. Yoon, M. Adams, E. D'Azevedo, and P. H. Worley, "Electrostatic gyrokinetic simulation of global tokamak boundary plasma and the generation of nonlinear intermittent turbulence," *ArXiv e-prints*, Jan. 2017.

[38] E. Endeve, C. Y. Cardall, R. D. Budiardja, and A. Mezzacappa, "Generation of magnetic fields by the stationary accretion shock instability," *The Astrophysical Journal*, vol. 713, no. 2, p. 1219, 2010.

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[40] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[41] U. Ayachit and et al., "Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis SC'16*. Piscataway, NJ, USA: IEEE Press, 2016, pp. 79:1–79:12.

[42] Y. Wang, G. Agrawal, T. Bicer, and W. Jiang, "Smart: A mapreduce-like framework for in-situ scientific analytics," in *SC'15*. IEEE, 2015, p. 51.

[43] M. Dorier and et al., "Damaris/viz: a nonintrusive, adaptable and user-friendly in situ visualization framework," in *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*. IEEE, 2013, pp. 67–75.

[44] P. Malakar and et al., "Optimal scheduling of in-situ analysis for large-scale scientific simulations," in *SC'15*. IEEE, 2015, pp. 1–11.

[45] J. C. Bennett and et al., "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *SC'12*. IEEE, 2012, pp. 1–9.

[46] C. Docan, M. Parashar, and S. Klasky, "Dataspaces: an interaction and coordination framework for coupled simulation workflows," *Cluster Computing*, vol. 15, no. 2, pp. 163–181, 2012.

[47] J. Dayal and et al., "Flexpath: Type-based publish/subscribe system for large-scale science analytics," in *2014 14th IEEE/ACM CCGrid*. IEEE, 2014, pp. 246–255.

[48] M. Dreher and T. Peterka, "Decaf: Decoupled dataflows for in situ high-performance workflows," Argonne National Lab.(ANL), Argonne, IL (United States), Tech. Rep., 2017.

[49] R. B. Ross, G. Amvrosiadis, P. Carns, C. D. Cranor, M. Dorier, K. Harms, G. Ganger, G. Gibson, S. K. Gutierrez, R. Latham, B. Robey, D. Robinson, B. Settlemyer, G. Shipman, S. Snyder, J. Soumagne, and Q. Zheng, "Mochi: Composing data services for high-performance computing environments," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 121–144, 2020. [Online]. Available: https://doi.org/10.1007/s11390-020-9802-0

[50] W. Liao, A. Ching, K. Coloma, A. Nisar, A. Choudhary, J. Chen, R. Sankaran, and S. Klasky, "Using mpi file caching to improve parallel write performance for large-scale scientific applications," in *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, Nov 2007, pp. 1–11.

[51] J. Lofstead, F. Zheng, S. Klasky, and K. Schwan, "Adaptable, metadata rich io methods for portable high performance io," in *In Proceedings of IPDPS'09, May 25-29, Rome, Italy*, 2009.

[52] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf, "Managing variability in the io performance of petascale storage systems," in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2010, pp. 1–12.

[53] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu, "On the root causes of cross-application i/o interference in hpc storage systems," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 750–759.

[54] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, Sept 2006.

[55] J.-l. Gailly, "gzip: The data compression program," 2016. [Online]. Available: https://www.gnu.org/software/gzip/manual/gzip.pdf

[56] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. Samatova, "Compressing the incompressible with isabela: In-situ reduction of spatio-temporal data," *Euro-Par 2011 Parallel Processing*, pp. 366–379, 2011.