Taming I/O Variation on QoS-Less HPC Storage: What Can Applications Do?

Zhenbo Qiao
New Jersey Institute of Technology
Newark, NJ, USA
zq38@njit.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

Jieyang Chen
Oak Ridge National Laboratory
Oak Ridge, TN, USA
chenj3@ornl.gov

***Abstract*—As high-performance computing (HPC) is being scaled up to exascale to accommodate new modeling and simulation needs, I/O has continued to be a major bottleneck in the end-to-end scientific processes. Nevertheless, prior work in this area mostly aimed to maximize the average performance, and there has been a lack of study and solutions that can manage I/O performance variation on HPC systems. This work aims to take advantage of the storage characteristics and explore application level solutions that are interference-aware. In particular, we monitor the performance of data analytics and estimate the state of shared storage resources using discrete fourier transform (DFT). If heavy I/O interference is predicted to occur at a given timestep, data analytics can dynamically adapt to the environment by lowering the accuracy and performing partial or no augmentation from the shared storage, dictated by an augmentation-bandwidth plot. We evaluate three data analytics, XGC, GenASiS, and Jet, on Chameleon, and quantitatively demonstrate that both the average and variation of I/O performance can be vastly improved using our dynamic augmentation, with the mean and variance improved by as much as 67% and 96%, respectively, while maintaining acceptable outcome of data analysis.**

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

I. Introduction

As high-performance computing (HPC) systems are being scaled up to exascale to accommodate new modeling and simulation needs, I/O has continued to be a major bottleneck in the end-to-end scientific processes. To allow science to be done more efficiently and effectively, there has been a multitude of research efforts addressing the I/O bottleneck across the hardware and software stack, including the device, system software and algorithmic level. Fundamentally, the I/O issue on HPC systems is attributed to the design philosophy that prioritizes the flops improvement, without fully balancing the ratio between compute and storage resources that are available on HPC systems [1]. For example, from the 27-petaflop supercomputer Titan [2] to the current 200-petaflop Summit [3] at Oak Ridge National Laboratory, the compute capacity is increased by nearly 10X, whereas the peak file system throughput remains approximately the same. The I/O challenge is further exacerbated by the exponential growth of data coming out of large-scale simulations that need to be stored and analyzed efficiently. As a result, HPC applications routinely suffer from subpar I/O performance, despite the unparalleled compute capabilities empowered by accelerators.

Overall, to characterize the I/O performance, not only the average I/O throughput and latency are relevant, but more broadly the consistency of I/O performance would matter to avoid over-stressing the storage system and unexpected job termination, as well as achieve improved user experience for interactive data exploration. Nevertheless, the majority of prior work in this area aimed to maximize the average performance, and there has been a lack of study and solutions that can manage I/O performance variation on HPC systems. A root cause of I/O variation on HPC systems is that large HPC storage systems do not provide explicit quality of service (QoS) mechanisms on a per application basis [4]. For example, the Lustre parallel file system on Theta [5] at Argonne National Laboratory is shared among all applications in a best-effort manner. As such, if a particular application consumes higher bandwidth, e.g., checkpointing more frequently on a storage service [6], the perceived bandwidth from other users will decrease commensurately. The rationale behind such QoS-less designs is that HPC systems typically have high parallelism (e.g., hundreds of thousands of cores), and designing a scheduler that can sustain a large amount of concurrent I/O requests while maintaining high performance is difficult. As a result, prior work in this area are limited to reducing, instead of completely eliminating, the variation via smart data placement [7], [8], [9] and explicit coordination [10] at the middleware level. Complementary to the prior work, this paper aims to further explore what we can achieve at the application level by co-designing data storage and analytics to manage the I/O variation, acknowledging that the I/O variation will continue to exist for the foreseeable future (e.g., on Summit). In particular, this work is driven by the following three observations.

First, HPC applications are shown to follow certain patterns

SC20, November 9-19, 2020, Is Everywhere We Are
978-1-7281-9998-6/20/$31.00

in the form of $I(C^xW)^*F$ [11], [12]. Herein I and F are the one-time initialization and finalization phases, respectively. The iterative phase contains C, the compute phase, and W, the periodic I/O phase. Since the I/O interference is a result of other concurrent applications competing for storage bandwidth, we can treat the interference as $\sum_i I_i(C_i^xW_i)^*F_i$, where the subscript denotes the i-th application. As such, we believe the noise pattern exists and can be learned and predicted.

Second, to lower the I/O overhead, using the reduced accuracy for data analysis is acceptable for science production, e.g., using lossy compression [13], [14] and reduced computational models [15] to make the problem more tractable. The rationale behind this is that scientific simulations are often over-resolved to ensure the stability of the calculation, and therefore a reduced representation of data may suffice for many, if not all, data analysis. Additionally, a lower accuracy may guide the subsequent data analysis to be more targeted in the spatial or temporal domains, thereby reducing the data movement cost. If there are interesting phenomenon observed in the low accuracy, they can be further augmented by fetching the residuals from the storage system.

Third, HPC storage has become increasingly deep with the addition of new memory devices (e.g., non-volatile memory) and storage layers (e.g., burst buffer). The separation between the local on-node and remote shared storage resources creates a natural performance isolation opportunity for applications - if an application desires performance consistency, it should stay away from accessing the shared storage. On the other hand, we recognize that the analysis data need to be further transformed to fully utilize the speed and capacity characteristics of the storage hierarchy, instead of being stored monolithically on the parallel file system for capacity.

Motivated by the observations above, the work aims to take advantage of the storage characteristics and adapt data analytics in a way that is interference-aware. In particular, we actively monitor and estimate the state of shared storage resources. If heavy I/O interference is imminent, data analytics will dynamically adapt to it by lowering the accuracy and performing less or no enhancement from the shared storage. In particular, this work makes the following contributions.

- We co-design data storage and data analytics to manage the I/O performance variation on HPC systems. We believe this work is among the first to explore application level solutions to deal with I/O interference. In particular, we leverage the fact that parallel applications on HPC systems follow certain patterns, and therefore the interference can be predicted, and data analysis over the reduced representation can still produce acceptable results.
- We propose a discrete fourier transform (DFT) based online interference monitoring and estimation scheme, exploiting the periodicity of I/O patterns in HPC applications, and demonstrate the efficacy of our approach.
- Based upon the estimated interference, we perform dynamic augmentation based upon an augmentation-bandwidth plot. This allows data analytics to trade accuracy for performance when there is interference.
- We experimentally demonstrate the effectiveness of our scheme on Chameleon [16] using realistic data analytics, XGC, GenASiS, and Jet. The results show that the I/O performance can be vastly improved with regard to both mean and variance, e.g., by 67% and 96% for Jet, while achieving acceptable analysis outcome.

The remainder of this paper is organized as follows. Section II discusses the background and motivation of this work. Section III presents our design of using refactorization and prediction to manage I/O variation. 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

The periodicity of I/O interference. HPC applications were shown to perform I/O in an iterative manner [11], [12]. Listing 1 shows a code snippet of XGC [17], one of the largest applications on Summit at Oak Ridge National Laboratory. During each iteration, XGC dumps hundreds of MBs to GBs to storage for checkpointing and post-processing. To understand the performance outcome as a result of interference, we co-run a data analytics that retrieves and analyzes 64 MB of data per iteration, alongside two other simulation I/O kernels that perform periodic checkpointing to the Ceph file system on Chameleon. In particular, the two simulations checkpoint 256 MB for every 10 and 20 seconds, respectively. For the sake of convenience, we configure the Ceph file system (version 13.2.6) with a single shared pool with two object store daemons (OSDs) (with one hard disk in each OSD), and measure the perceived I/O throughput of data analytics in Fig. 1. Due to the periodicity of the simulations, the perceived bandwidth of the data analytics also exhibits a periodic pattern, and the loss of performance is a result of bandwidth consumption from the simulations.

```
do istep=1,sml_mstep
   ......
   if (mod(sml_gstep,sml_restart_write_period)==0)
    then
      call monitor_start(RESTART_WRITE_)
      call restart_write(spall) !periodic checkpoint
      call monitor_stop(RESTART_WRITE_)
   endif
enddo !end of main loop
```

Listing 1: Checkpoint routine in XGC.

QoS in HPC file systems. State-of-the-art HPC file systems, e.g., Ceph, Lustre, Spectrum Scale and OrangeFS, have shown very high performance as well as scalability. However, they have provided limited or no support towards application performance guarantee. Table I lists a few major file systems deployed at large HPC centers along with their QoS mechanisms.

In particular, there has been effort in Ceph [18] to integrate *dmClock* [19], a distributed resource allocation algorithm, to

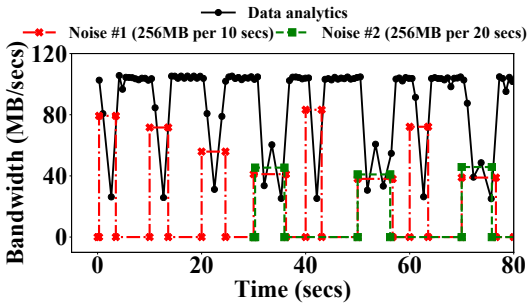


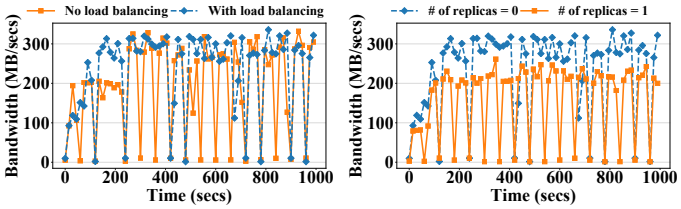
Fig. 1: Performance degradation of data analytics as a result of interference (Ceph). Here we co-run a data analytics that retrieves and analyzes 64 MB of data per iteration, alongside two other simulation I/O kernels that perform periodic checkpointing of 256 MB every 10 secs and 20 secs to Ceph on Chameleon.

TABLE I: QoS in HPC file systems.

File system	HPC System	QoS mechanism
Ceph (13.2.6)	CERN cluster	None. A new mechanism is under development using *dmclock*.
Lustre (>2.6)	Theta, Cori	Token bucket filter. However, it can only throttle I/O performance, without providing performance guarantee. Can only limit RPC rate, instead bandwidth and IOPS.
Spectrum Scale (5.0.4)	Summit	QoS can only be controlled using *mmlsqos* for each storage pool, instead of for individual applications.
OrangeFS (2.9.7)	AWS	None.

provide QoS. However, a hardened solution with *dmClock* is not available as of now, and in production environments applications can somewhat reduce the performance variation through load balancing. Ceph uses CRUSH [20] to distribute data among storage devices. To avoid load imbalance, it distributes new data randomly, which intuitively should alleviate the interference to some degree. However, this distribution strategy is primarily capacity driven, instead of bandwidth driven, and therefore plays a limited role in reducing the I/O variation. In Fig. 2a, we test the effect of load balancing using CRUSH with replication disabled. To achieve no load balancing, we overwrite the same object across runs. As such, the data placement is identical across the run. It is shown that that with load balancing, the interference still exists but appears less frequently. Fig. 2b further examines the performance variation with replication enabled to achieve fault tolerance. With one replica for each write, the interference appears more frequently than no replication and the effect of load balancing is diminished.

Lustre is a popular storage solution on extreme-scale systems and has demonstrated success in managing petabytes of data for hundreds of thousands of clients. Despite its recent improvement using a token bucket filter to provide QoS [21], it suffers two major weaknesses. First, it can only throttle I/O performance through limiting the RPC rate, as opposed to providing performance guarantee for bandwidth and IOPS that users are mostly concerned with [22]. Second, there is no global QoS management that coordinates all storage devices.



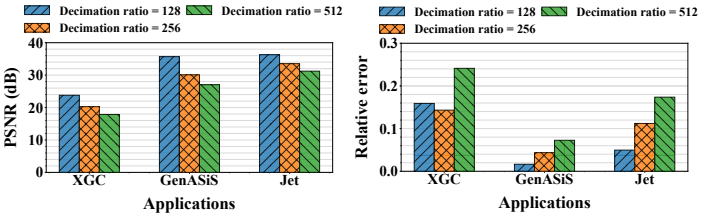
(a) Impact of load balancing (# of replicas = 0).

(b) Impact of replication (with load balancing enabled).

Fig. 2: Performance variation on Ceph. Herein the noise writes 1 GB every 60 secs, and the data analytics retrieves and analyzes 64 MB of data per iteration.

Therefore, the QoS management in production is cumbersome and ad hoc [22]. Meanwhile, Spectrum Scale (formerly GPFS) provides QoS through *mmlsqos* [23] on a per storage pool basis, and there are no mechanisms in place to provide application-level performance guarantee. Last, OrangeFS [24] is a user-level file system that is specifically designed to optimize MPI-IO in a parallel environment. OrangeFS relies on the lower level file system to achieve QoS.

Impact of reduced accuracy to data analytics. A key premise of this work is that a reduced representation of data can be useful and achieve reasonable outcome for data analysis. This is due to the fact that simulations are often over-resolved to ensure the stability of the calculation after many iterations, and therefore the resulting accuracy may be excessive for many data analytics. Fig. 3a measures the peak signal-to-noise ratio (PSNR) of the data products of three applications, XGC, GenASiS, and Jet, versus various reduction ratio. Herein, we use the uniform decimation (Section III-B) to produce the reduced representation. For example, at the decimation ratio of 2, we retain the first data point for every two data points. It is found that the degradation of PSNR is rather insignificant after decimation. Fig. 3b further shows the deviation of analysis outcome from the ground truth. For XGC, the analysis conducted is blob detection and we measure the error regarding the number of blobs detected. For GenASiS, we perform the 2D rendering of the velocity of core-collapse and measure the structural similarity index (SSIM) [25]. For Jet, we measure the error regarding the area with high pressure area. More details regarding the data analytics can be found in Section IV-A. Even with the decimation ratio of 512, the relative error is still within 25%.



(a) PSNR.

(b) Analysis error.

Fig. 3: Accuracy of using the reduced representation.

HPC storage hierarchy. HPC storage systems have become increasingly deep with the emergence of new memory systems,

TABLE II: Storage hierarchy on large systems. Herein we describe the characteristics of only two tiers. The notion of *local* can be either node local or cluster local. We broadly classify the intensity of interference to *none*, *medium*, and *high*. For Cori, since the high tier is only accessible from a single system, instead of from multiple systems, we mark the interference as *medium*.

System	High tier		Low tier	
	Storage	Interference	Storage	Inteference
Cori	Cluster local HDD, Lustre	Medium	Shared Lustre	High
Summit	Node local NVMe, XFS	None	Shared Spectrum Scale	High
Theta	Node local SSD, ext3	None	Shared Lustre	High

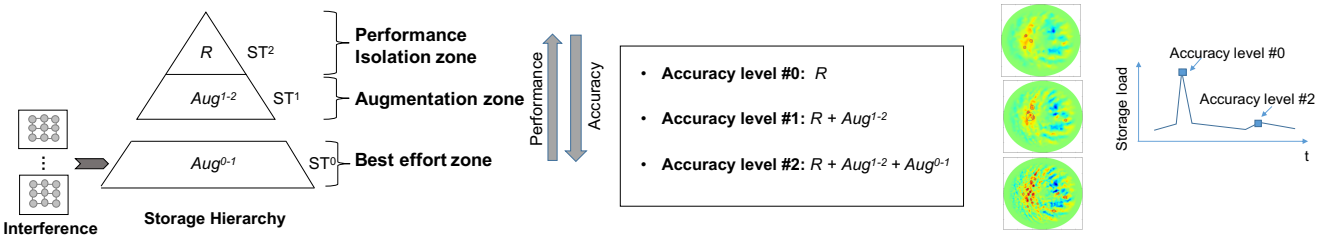


Fig. 4: An illustration of the proposed work. Herein we assume the interference is over the shared ST^0. The data analytics can perform in three accuracy levels. If there is interference experienced, the data analytics can adapt and lower the accuracy, so that there are less or no data retrieved from ST^0.

such as die-stacked memory and NVRAM, as well as the addition of burst buffer to the storage landscape. Table II describes the storage tiers (with only two tiers due to the space constraint) on large HPC systems. A critical gap in the current scientific data management is that the analysis output from simulations is monolithic and large in volume, and therefore can only be accommodated from the disk based parallel file system for persistent storage. We believe data refactorization (Section III-B) that transforms data to a reduced representation in conjunction with a series of residuals is needed to fully leverage the hierarchical nature of HPC storage, and allow users to trade accuracy for performance. More importantly, the high storage tiers are often designed to the local (e.g., node or cluster local) and the low tiers tend to be shared. Such a design provides a natural opportunity for performance variation mitigation. When there is interference observed on the shared storage, data analytics can access less or refrain from accessing the share storage to reduce the I/O variation.

III. Design

A. Overview

Fig. 4 provides an overview of the proposed work. The key idea is to learn the pattern of interference, and dynamically adapt the accuracy of data analytics by taking advantage of the refactored data distributed across the storage tiers. By accessing less or no data from those shared storage tiers where there is interference, the performance variation can be mitigated. For the convenience of the discussion, we list the notations in Table III. Without the loss of generality, we consider a storage system with three tiers $\{ST^0, ST^1, ST^2\}$, with ST^0 being the slowest tier and ST^2 being the fastest tier. As aforementioned, high storage tiers on HPC systems, e.g., ST^2, tend to be node local and will be reserved alongside compute resources once a job is submitted. Therefore performance isolation can be achieved using these tiers, referred to as *performance isolation zone*. In contrast, for those cluster local storage, such as Cori scratch [26], performance cannot be fully guaranteed, but the interference will be relatively low as compare to the remote shared storage. These tiers will be accessed only if the accuracy of data analytics needs to be enhanced, here referred to as *augmentation zone*. If the full accuracy is ultimately desired, those lower tiers (e.g., ST^0), referred to as *best effort zone*, will be accessed.

TABLE III: A list of notations.

Variable	Description
$Aug^{l-(l+1)}$	the augmentation that elevates the accuracy from level $l+1$ to l.
$Aug_x^{l-(l+1)}$	the x-th data point in the augmentation that elevates the accuracy from level $l+1$ to l.
$Bandwidth_{low}$	lower bandwidth threshold in the augmentation-bandwidth plot. If the predicted bandwidth is lower than this, storage is deemed to be highly congested.
$Bandwidth_{high}$	higher bandwidth threshold in the augmentation-bandwidth plot. If the predicted bandwidth is higher than this, storage is deemed to be lightly congested.
$Bandwidth_{predicted}$	predicted bandwidth at the target step.
d^l	decimation ratio of level l.
$f(\cdot)$	augmentation-bandwidth plot.
$Estimate(\cdot)$	A linear function that provides prolongation and correction between levels
L^l	the l-th level data representation.
L_x^l	the x-th data point in l-th level data representation.
ST^l	the l-th storage tier.
$thresh$	threshold of frequency amplitude.
R	reduced data representation.

Assume that we offer three levels of accuracy to data analysis and the associated data representations are $\{L^0, L^1, L^2\}$ with L^0 being the original data. To take advantage of the three tiers, the analysis data produced by simulations need to be further refactored in the form of $\{R, Aug^{1-2}, Aug^{0-1}\}$, where $R = L^2$ is the reduced representation along with augmentations Aug^{1-2} and Aug^{0-1}. In particular, $Aug^{l-(l+1)}$ elevates the accuracy from L^{l+1} to L^l. As such, three levels of accuracy

can be offered to data analysis: 1) the low accuracy $L^2 = R$, 2) the medium accuracy using $L^1 = R + Aug^{1-2}$, and 3) the high accuracy $L^0 = L^1 + Aug^{0-1} = R + Aug^{1-2} + Aug^{0-1}$. Note that "+" is a linear operator that builds a higher accuracy via interpolation and correction (Section III-C). When the interference over shared storage tier ST^0 is predicted to be low, data analytics can retrieve $\{R,\ Aug^{1-2},\ Aug^{0-1}\}$ to construct the high accuracy. Otherwise, data analytics can retreat to the augmentation zone and fetch $\{R,\ Aug^{1-2}\}$. If performance isolation is desired instead, e.g., for those interactive analytics, the data analytics can stay within the performance isolation zone by fetching $\{R\}$ only.

At the core of our method, we need to capture and estimate the interference. To this end, we collect the performance history of data analytics *online* at each iteration and use it for prediction. This is done without either proactively probing the shared storage (e.g., by periodically issuing I/O requests to storage and measuring the performance), or retrieving those internal performance counters maintained by the file system. The former method is intrusive and costly for a storage system with a large number of devices, while the latter is typically unavailable to the general users due to security concerns. Herein, driven by the periodic pattern of HPC applications, we use a DFT-based approach to extract the frequency components to estimate the interference from other applications. The estimation is done and updated online as data analytics progresses, and can adapt when the pattern of interference changes (Section IV-C).

B. Refactorization

The goal of refactorization is to provide a series of data representations to satisfy various data analysis. We comment that a user may not know the analysis to be performed in advance (a.k.a exploratory data analysis) when data needs to be placed on storage. Therefore, our work adopts a generic technique that decomposes data into various levels to allow users to pick the level of data and augment it if needed. To this end, we transform a dataset into its reduced representation alongside a set of augmentations that can be readily mapped to the storage hierarchy. *Carefully note that data are refactored once but will be retrieved many times by data analytics, and therefore the benefits to the data analysis can amortize the additional cost in data refactorization, as demonstrated by prior work [27].* The refactorization process consists of decimation and augmentation construction as described below.

1) Decimation: The goal of decimation is to produce a set of representations $\{L^0,\ L^1,\ L^2\}$. This work adopts a simple refactoring technique based upon uniform decimation. In particular, to build a reduced representation L^l, $l = 1, 2$, we linearize the data into a 1D array and retain every d^l-th element starting from the first element, where d^l is the decimation ratio, a parameter that users can adjust so that L^l satisfies the capacity requirement. The time complexity of uniform decimation is $O(N)$, which is more efficient than other advanced techniques, such as the edge collapse based topological decimation [28], while achieving similar outcome for data analysis.

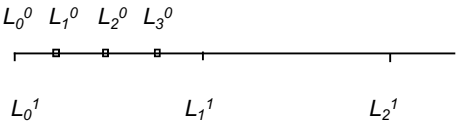


Fig. 5: An example of the augmentation construction for Aug^{0-1} for a decimation ratio of $d^1 = 4$.

2) Augmentation Construction: After decimation, we further construct the two augmentations $\{Aug^{1-2}, Aug^{0-1}\}$. The role of Aug^{1-2} and Aug^{0-1} during data analysis is to elevate the accuracy from R to L^1 and from L^1 to L^0, respectively, when the interference is predicted to be low. Intuitively, an augmentation is the difference between the adjacent levels of representations. We use L_x^l and $Aug_x^{l-(l+1)}$ to denote the x-th data element in L^l and $Aug^{l-(l+1)}$, respectively. Eq. (1)–(2) specify the augmentation construction mathematically.

$$Aug_x^{1-2} = L_x^1 - Estimate(L_i^2, L_{i+1}^2) \quad (1)$$

$$Aug_x^{0-1} = L_x^0 - Estimate(L_i^1, L_{i+1}^1) \quad (2)$$

$Estimate(\cdot)$ is a linear function that involves the prolongation and correction of the two neighboring data points L_i^{l+1} and L_{i+1}^{l+1} from level $l+1$ to level l, leveraging that the two levels are expected to be highly correlated, similar to the idea of multigrid [29]. Fig. 5 shows an example of constructing Aug^{0-1} for a decimation ratio of 4. In particular, $Aug_0^{0-1} = L_0^0 - L_0^1 = 0$, $Aug_1^{0-1} = L_1^0 - Estimate(L_0^1, L_1^1) = L_1^0 - (\frac{3}{4}L_0^1 + \frac{1}{4}L_1^1)$, $Aug_2^{0-1} = L_2^0 - Estimate(L_0^1, L_1^1) = L_2^0 - (\frac{1}{2}L_0^1 + \frac{1}{2}L_1^1)$, and $Aug_3^{0-1} = L_3^0 - Estimate(L_0^1, L_1^1) = L_3^0 - (\frac{1}{4}L_0^1 + \frac{3}{4}L_1^1)$. We note that the augmentation may be further reorganized so that it can be later retrieved efficiently from a particular storage tier. For example if Aug^{0-1} is mapped to hard disks, the data points will be further sorted by their magnitude so that data points with high corrections can be augmented first during a partial augmentation and retrieved in large chunks to reduce the disk seek overhead.

C. Interference Estimation and Mitigation

The goal of interference estimation is to predict the characteristics (frequency, intensity, phase) of interference, so that we can determine the augmentation to be conducted from the shared storage at a given timestep t_i. The stronger the interference is, the less the augmentation will be fetched. Fig. 6 illustrates the steps involved in interference estimation and mitigation. The data analytics in the context of this work are those that analyze the time evolution of a physical phenomena in a simulation, typically spanning thousands of steps, and therefore is iterative in nature. The idea of interference estimation is to collect the performance of data analytics in every N (e.g., 48) steps as it progresses, and perform DFT to extract those frequency components that are higher than a given threshold $thresh$. In doing so, the interference from other simulations can be captured, while those random noise, e.g., as a result of code compilation, user issued shell

commands, and operating system noise, etc., can be filtered out. Subsequently, we perform inverse DFT to convert the measurements back to the time domain to obtain the available bandwidth $Bandwidth_{predicted}$ at a target step t_i.

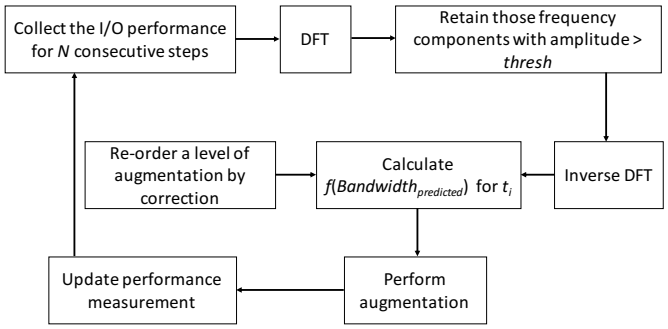


Fig. 6: Interference estimation and mitigation. Note that $f(\cdot)$ is augmentation-bandwidth plot.

Next, $Bandwidth_{predicted}$ is mapped to a detailed augmentation strategy, e.g., the degree and the regions to augment, through an augmentation-bandwidth $f(\cdot)$ as shown in Fig. 7. In general, the augmentation strategy is application dependent, and in this work we consider a generic correction based augmentation strategy, where we identify and prioritize the augmentation for those regions with high corrections. The degree of augmentation under interference is proportional to $Bandwidth_{predicted}$, and can be determined by a simple augmentation-bandwidth plot, as shown in Fig. 7. At t_i, if $Bandwidth_{predicted} \leq Bandwidth_{low}$, the shared storage is deemed to be highly congested, and therefore no augmentation will be done to avoid further overloading the storage system, i.e., $f(Bandwidth_{predicted}) = 0$. Such a design is chosen given that the disk performance becomes rapidly degraded once the load is high enough [30]. If $Bandwidth_{predicted} \geq Bandwidth_{high}$, the shared storage is deemed to be in low congestion, and hence a full augmentation will be performed, i.e., $f(Bandwidth_{predicted}) = 1$. Otherwise, a proportional augmentation in the form of $f(Bandwidth_{predicted}) = k \cdot Bandwidth_{predicted} + b$, where k and b are coefficients, will be done. As such, the number of data points to be augmented is proportional to the available bandwidth predicted at t_i, and the set of data points to be augmented is based upon the correction at each point - a data point with a larger correction will be augmented earlier than one with a smaller correction, until we reach the augmentation degree. We note that the coefficient k as well as the bandwidth thresholds affect the sensitivity of augmentation to the available bandwidth (studied in Section IV-C). Note that our approach is applicable to storage systems that have different performance characteristics (e.g., peak bandwidth, variation), and the only change that a user may make is the parameter of the augmentation-bandwidth plot (i.e., k and b). For example, as a rule of thumb, for a storage system that experiences higher interference, one can lower k to reduce the augmentation degree to reduce the performance variation.

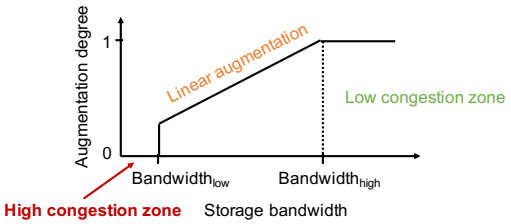


Fig. 7: Augmentation-bandwidth plot. If the predicted bandwidth is lower than $Bandwidth_{low}$, storage is deemed to be in congestion, and therefore no augmentation is performed. If the predicted bandwidth is higher than $Bandwidth_{high}$, storage is deemed to be in low congestion, and therefore a full augmentation is performed. Otherwise, a partial augmentation is done.

Once the augmentation is fetched from a storage tier, the target representation can be constructed by Eq.(3)-(4).

$$L_x^1 = Aug_x^{1-2} + Estimate(L_i^2, L_{i+1}^2) \quad (3)$$

$$L_x^0 = Aug_x^{0-1} + Estimate(L_i^1, L_{i+1}^1) \quad (4)$$

Once the augmentation is completed at t_i, we update the performance measurement, move the sliding window forward, and proceed with a new around of estimation for t_{i+1}. Note that the complexity of DFT and inverse DFT is O(Nlog(N)). In the event of high congestion where there is no I/O performed from the shared storage, we instead use the predicted performance for the new round of estimation.

IV. Evaluation

A. Experimental Setup

In this work, we use Chameleon, a reconfigurable experimental platform at University of Chicago and Texas Advanced Computing Center, to set up the test environment. We configure one client to run data analytics, and three clients to inject noises concurrently mimicking the interfering applications, as shown in Fig. 8. We use Ceph (13.2.6) as the underlying object storage system that is configured with two storage nodes. Based upon the hardware that is available, we build a two-tier storage system with the fast tier being node local SSDs, and the slow tier being hard disks (HDDs). In particular, the latter is configured as a OSD pool shared between data analytics and noises. Note that large-scale production systems may have more storage capacity and throughput than our testbed. However, this does not translate to less performance variation, since they potentially may have even more sources of interference. As demonstrated in early work [7], [8], [9], [10] from other investigators, the interference on large HPC systems, such as Titan, can be severe and can degrade the I/O performance by an order of magnitude. In general, we anticipate the interference on large systems to be heavy due to the fact that a single storage device could be accessed by many processors. Furthermore, the ratio of compute to storage is generally increasing for HPC systems [1], and therefore the intensity of interference may further increase. By default, the decimation ratio used to construct the reduced representation is 1024, unless otherwise specified. Table IV details the hardware

configuration and Table V describes the noises we inject to Ceph. By default, we inject three concurrent noises to Ceph, with the first one writing 256 MB (*small*) every 80 seconds, the second writing 512 MB (*medium*) every 60 seconds, and the third writing 768 MB (*large*) every 100 seconds respectively. The reason to choose these high-velocity high-volume noises as default is to study how well our scheme can perform under high interferences, a scenario that particularly impacts interactive data analytics, e.g., visualizations, while the less intensive noise setup and the relationship between noise and data analytics frequency are also studied in Section IV-D1 and IV-D2 for comparison.

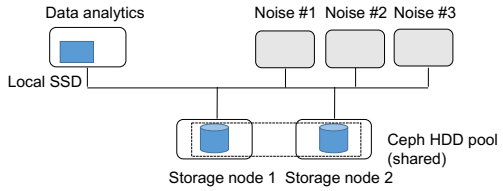


Fig. 8: Test environment.

TABLE IV: Node configuration.

Node type	Client node	Storage node
Processor	Intel Xeon 2.60 GHz	Intel Xeon 2.30GHz
Number of cores	12	10
Storage device	Samsung 240G SATA SSD	Seagate 2TB 7200RPM SAS3 HDD

TABLE V: Noise injected to the shared storage.

Noise	Period	Checkpoint Size
Noise #1	80 secs	256 MB
Noise #2	60 secs	512 MB
Noise #3	100 secs	768 MB

Table VI shows the three data analytics we used for evaluation. We note that the datasets examined here are the periodic analysis outputs from the simulations (as opposed to checkpoints) which will be retrieved by data analytics for post-processing and visualization. For XGC, we apply this work to the so-called *blob detection* [31], [17] for data produced by the fusion XGC simulation. These blobs are plasma regions with high electrostatic potential. When developed near the edge of the fusion reactor, it can quickly dissipate the confined particle that ultimately causes catastrophic disruptions. This data analytics is to examine the characteristics of the *dpot* quantity, which measures how the electric potential deviates from background. The mesh of the XGC dataset consists of 9,984,141 triangles. GenASiS [32] 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 of the dataset consists of 10,534,050 triangles. Jet studies and analyzes the interaction of liquids with surfaces under certain boundary conditions. This data analytics examine the *pressure* near the front of a fighter jet. The mesh of the entire jet consists of 2,278,854 triangles.

TABLE VI: Data analytics used in this paper.

Name	Application Area	Data Analytics
XGC	Fusion	Blob detection
GenASiS	Astrophysics	Rendering
Jet	CFD	Extracting high-pressure area

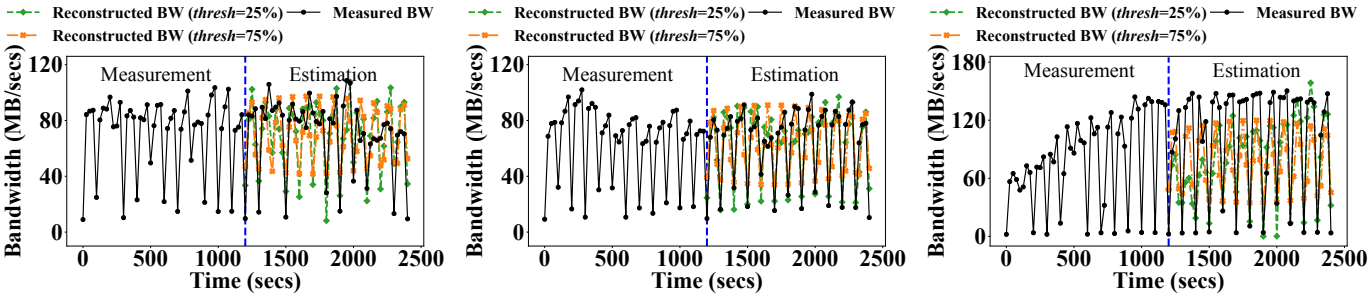
B. Justification of Methodology

As the very first step, Fig. 9 demonstrates the viability of our DFT-based approach to capture the characteristics of interference. We collect the I/O performance of XGC, GenASiS, and Jet for a period of 20 mins, and use the performance data to predict the available storage bandwidth for the subsequent 20 mins. For the sake of comparing the prediction to the actual performance, we also record the storage performance during the target prediction window. As aforementioned, we extract those frequency components that are deemed to be high in amplitude, i.e., with $thresh$=25% and 75% of the maximum amplitude, respectively, and run the DFT-based algorithm. It is clear that the results match the actual measurements well with regard to the amplitude, frequency as well as phase, despite the insignificant frequency components being discarded. Note that a higher $thresh$ will result in fewer frequency components being extracted, thereby affecting the amplitude measured in the time domain. This enables us to accurately estimate when and how much the interference will be.

On the other hand, lowering the accuracy is acceptable and should still produce meaningful, if not identical, outcome for data analysis. As a case study, Fig. 10 provides a quantitative analysis of blob detection results for XGC. Herein, we manually vary the augmentation degree and assess the total number of blobs detected, average blob diameters, total blob area, and blob overlapping ratio. It is shown that even without augmentation, each metric is still acceptable under such an extreme decimation ratio. For example, the total blob area detected only deviates from the ground truth by 20% using less than 0.1% of the original data. As we further increase the augmentation degree, all metrics improve and the false positives of blobs reduce.

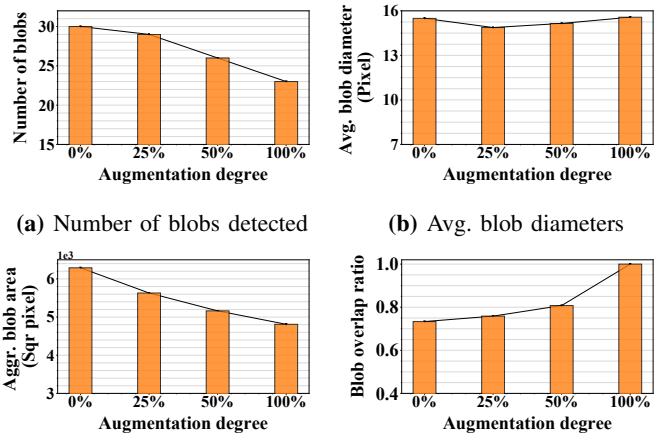
C. Dynamic Augmentation

We next evaluate the effectiveness of dynamically adjusting the degree of augmentation dictated by the augmentation-bandwidth plot. In Fig. 11, we measure the performance of data analytics in the following three cases: **case 1):** there is no prediction and a full augmentation is done by fetching data from the shared Ceph HDD pool, here referred to as *best effort*; **case 2):** the interference is predicted based upon the performance data collected during the first 20 mins, and the augmentation degree is determined by the augmentation-bandwidth plot, here referred to as *dynamic augmentation*. We also vary the coefficient k and b, and study their impact; and **case 3):** we only retrieve the reduced representation from SSDs and perform no augmentation to achieve performance isolation, referred to as *performance isolation*, to show the best possible scenario for variation reduction.



(a) XGC. **(b)** GenASiS. **(c)** Jet.

Fig. 9: DFT-based interference estimation. Herein we use the performance measurements during the first 20 mins, as shown in the left plane of each plot, to predict the subsequent 20 mins, as shown in the right plane.



(a) Number of blobs detected **(b)** Avg. blob diameters

(c) Blob area **(d)** Blob overlapping ratio

Fig. 10: A quantitative evaluation of blob detection.

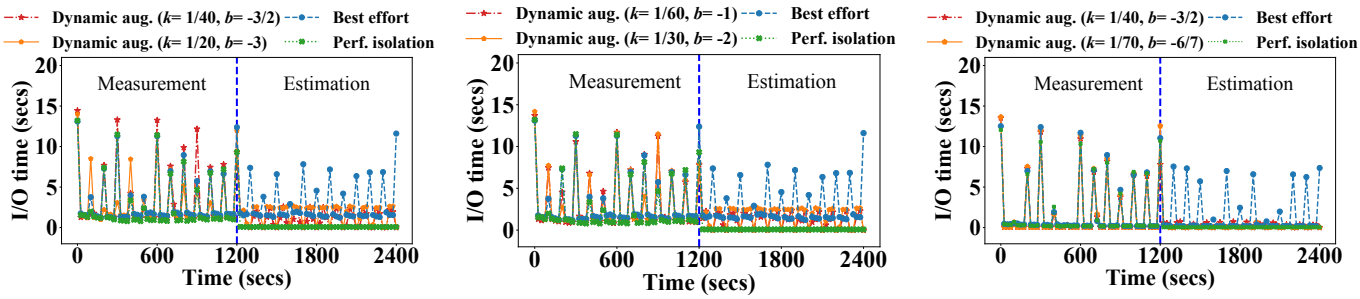
The results in Fig. 11 demonstrate that the dynamic augmentation can avoid the interference very effectively. We note that with dynamic augmentation, there are still data retrieved from the shared storage to improve the accuracy of data analytics when the interference is not at peak. As a result, the variation is much lower but still exists. Meanwhile, in the case of performance isolation, we only retrieve the reduced representation from the node local SSDs and therefore the performance is consistent. On the other hand, coefficient k affects the aggressiveness of augmentation - a larger k will result in a higher augmentation under the same intensity of interference, thus being more susceptible to I/O performance degradation. Fig. 12 further calculates the mean and variance of the I/O performance. As compared to best effort, it is evident that not only the average performance, but also the consistency of I/O is significantly improved for dynamic augmentation. For example, for Jet, the mean and variance of I/O time is reduced by 67% and 96%, respectively, as compared to best effort.

Meanwhile, the quality of data with regard to PSNR is assessed in Fig. 13. Here the original data are deemed to the reference signal and PSNR is calculated as the ratio of the original data to the error as a result of partial or no augmentation. The performance isolation yields the lowest quality, and as the coefficient k increases, the quality improves with dynamic augmentation. Similarly, Fig. 14 evaluates the outcome of data analytics. For XGC, we evaluate the deviation of blob features with regard to the number of blobs, average blob diameter, total blob area, and blob overlapping ratio from the ground truth. For GeASiS, we calculate the structural similarity index (SSIM) [25] and Dice's coefficient [33] to assess the deviation of the rendering results. The SSIM is a perceptual metric that quantifies image quality degradation caused by the loss of accuracy, and the Dice's coefficient is a statistical tool to measure the similarity between two images. For Jet, we evaluate the high pressure area and the total force exerted on the high pressure area. It is evident that the dynamic augmentation provides good trade-offs between performance isolation and best effort so that domain scientists can make more informed decisions between accuracy and performance when there is interference.

In reality, the noise pattern is not static and will change when a new simulation starts running or an existing simulation terminates. Therefore, it is important to understand how quickly our algorithm can adapt to the change of interference. Fig. 15 shows the relative prediction error, measured as $\frac{|Bandwidth_{predicted} - Bandwidth_{real}|}{Bandwidth_{real}}$, when a new simulation starts dumping 512 MB every 75 secs starting around 1600 sec. Herein our DFT-based estimate is performed upon a sliding window of 600 secs. Initially, the prediction error is high since the sliding window still involves samples collected prior to 1600 sec when the noise pattern did not change. However, as the sliding window moves forward in time, the overall trend is that the prediction error is gradually reduced. After 2200 sec, all samples in the sliding window reflect the new interference pattern, and therefore, the prediction error is substantially reduced. Overall, it takes the length of the sliding window for our algorithm to adapt to new interference pattern.

D. Parametric Study

1) Intensity of Interference: To understand the impact of interference, we evaluate how well the dynamic augmentation performs under lighter interference. Table VII lists the parameters of the lighter noises, and Fig. 16 shows the mean and variance of the I/O performance of XGC. For best effort, the performance variation is sensitive to the intensity of interfer-



(a) XGC. **(b)** GenASiS. **(c)** Jet.

Fig. 11: DFT-based interference estimation and dynamic augmentation. The noises we injected here are listed in Table V. The legend of *best effort* denotes the case that there is no prediction and the full augmentation is done by fetching the augmentation from the shared storage. The legend of *dynamic aug.* denotes the case that there is augmentation and the degree of it is determined by the augmentation-bandwidth plot. The legend of *perf. isolation* denotes the case that we only retrieve the reduced representation from the node local SSDs, and therefore there is no interference. Here we use the performance data collected during the first 20 mins to predict the subsequent 20 mins. Note k and b are coefficients of the augmentation-bandwidth plot.

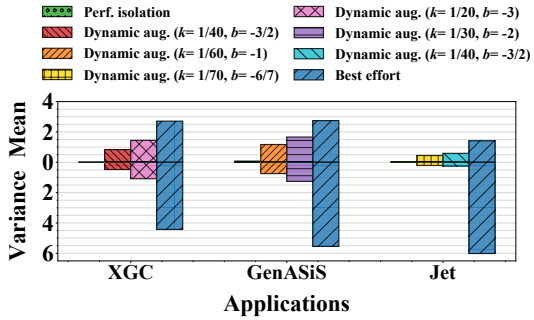


Fig. 12: I/O time statistics (collected from 1200 to 2400 secs). The upper part shows the mean, while the lower part shows the variance.

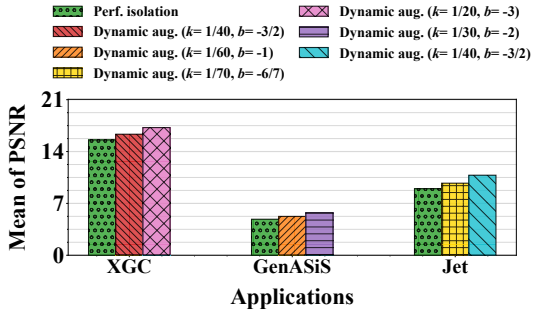


Fig. 13: Assessing data quality using PSNR.

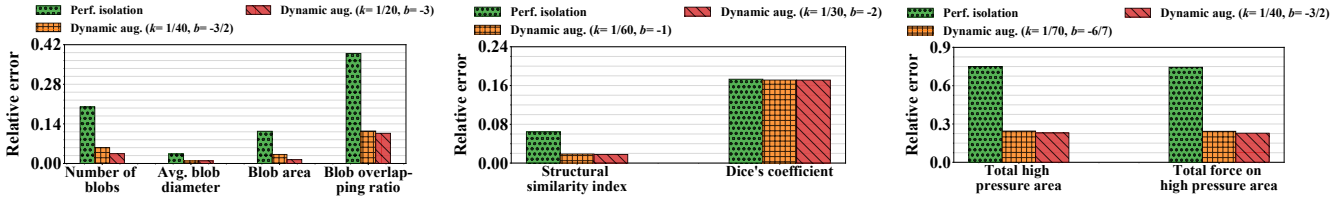
ence and reduces by 80% as the interference becomes lighter. With dynamic augmentation, the variation is less sensitive to the noise intensity, since the degree of augmentation is adjusted on-the-fly based upon the augmentation-bandwidth plot, thus obtaining higher accuracy when the interference is lighter.

TABLE VII: Small noise injected to the shared storage.

Noise	Period	Checkpoint Size
Noise #1	80 secs	128 MB
Noise #2	60 secs	256 MB
Noise #3	100 secs	512 MB

2) I/O period of data analytics: Given the design that we collect the performance history of data analytics at each iteration to make estimation, the frequency of data analytics would matter - the less frequently the data analytics performs I/O, the less information we can capture for the storage state. Overall, to fully capture the noises on the storage system, the I/O frequency of data analytics, f_a, needs to be greater than twice the highest frequency of noise f_{max}, i.e., $f_a > 2f_{max}$, a.k.a the Nyquist frequency, as per the Nyquist-Shannon sampling theorem [34]. Therefore, for the three noises we inject with the period of 80 secs, 60 secs, and 100 secs, the period of data analytics should be less than 30 secs in order to fully capture all three noises. Otherwise, as the sampling period of data analytics drifts higher away from 30 secs, more frequency components related to the noises will get lost during DFT, and thus the less effective the estimation will be. Fig. 17 illustrates the impact of the period of XGC data analytics. In particular, Fig. 17a measures the impact of period on the I/O performance. For the period of 25 secs, since it is less than the Nyquist frequency, it has the best outcome of performance estimation, thus yielding the lowest mean and variance of I/O time. Once the period is increased beyond 30 secs (thus the frequency is lower than the Nyquist frequency), the performance estimation becomes less accurate and therefore the variation increases. In Fig. 17b, we calculate the average PSNR vs. the I/O period. As the period increases, the ability to capture the noises becomes lower, and therefore the estimated bandwidth becomes increasingly higher than what is actually available on the storage system. This results in higher augmentation degree, and in turn higher PSNR.

3) $Bandwidth_{low}$ and $Bandwidth_{high}$: To further understand how the bandwidth threshold would impact the I/O performance of XGC data analytics, we test different combinations of $Bandwidth_{low}$ and $Bandwidth_{high}$ in Fig. 18. In particular, at $Bandwidth_{low} = 60$, when $Bandwidth_{high}$ increases from 90 to 120, it essentially reduces the coefficient k. Therefore, under the same interference, the latter would retrieve less augmentation, thus resulting in lower mean and variance. The combination of $Bandwidth_{low} = 30$ and $Bandwidth_{high} = 120$ yields further higher I/O time and variation as a result of larger margin for augmentation.



(a) XGC. **(b)** GenASiS. **(c)** Jet.

Fig. 14: Assessing data quality with regard to the outcome of data analytics.

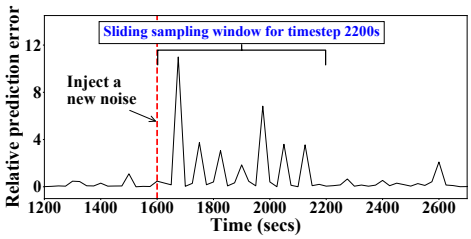


Fig. 15: Relative prediction error when injecting a new noise at 1600 secs. The new noise writes 512 MB every 75 seconds.

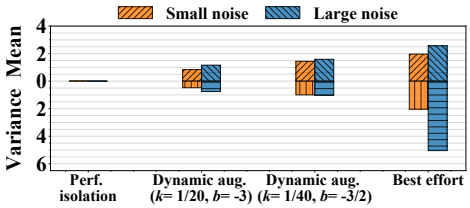
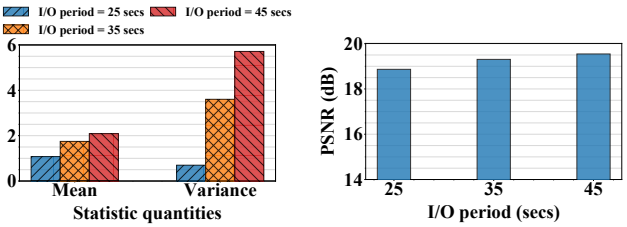


Fig. 16: Small vs. large noise (XGC).



(a) I/O time. **(b)** Average PSNR.

Fig. 17: Impact of I/O period of data analytics (XGC).

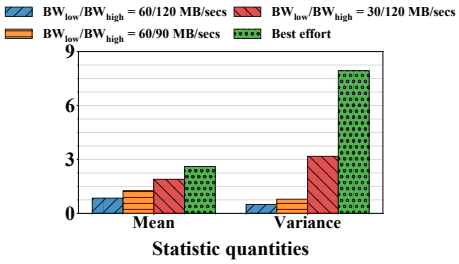


Fig. 18: Impact of $Bandwidth_{low}$ and $Bandwidth_{high}$ (XGC).

V. Related Work

Scientific data management has been identified as one of the top research challenges in exascale computing [35]. It is well recognized that the major steps in scientific processes, including data storage, analysis and visualization, can suffer greatly from the worsening I/O bottleneck. To accelerate knowledge discovery on HPC systems, there have been a multitude of efforts to address this issue, ranging from enhancing the HPC I/O systems, providing in-memory computing for data analytics, to reducing data on HPC systems. We next discuss each of these areas in great detail.

To significantly simplify and optimize data management on large-scale systems, new data processing frameworks, such as ADIOS [36] and Mochi [37], were proposed to allow for specialized methods to be pluggable for a diverse set of platforms, while maintaining simple declarative APIs for applications. Meanwhile, various optimizations and techniques have been proposed to further scale up I/O performance, with the central idea of better organizing data and utilizing the persistent storage [38], [39], [40]. However, most of these prior work do not consider the consistency of performance and are focused on improving either average or peak I/O performance. This results in a lower system utilization and subpar user experience for science production. Very recently, the issue of performance variation on large HPC systems has started to gain interest from the research community. Prior work [8], [9], [7] take an opportunistic approach to handling storage interference by offloading I/O requests from busy to idle storage devices. CALCioM mitigates interference through a cross-application coordination. However, it forces all applications on a system to use CALCioM APIs, and therefore is limited in wide adoption. Prior work [4] further provides in-depth analyses for the root cause of I/O interference. By and large, these work are mostly middleware level solutions. In this work, we believe there are further research opportunities in the application space to reduce the performance variation via trading accuracy for performance. Further, Gong et al. [41] attribute the network performance to be the combined effect of the constant and volatility components, and the goal was to only identify the constant component. In contrast, our goal is to estimate the combined effect which will affect the data analytics performance. Maricq et al. [42] estimate the number of times an experiment needs to be ran to achieve a certain confidence level. This work by itself does not reduce the performance variation of a system. In contrast, the goal of our work is to estimate the I/O performance and then reduce the performance variation of data analytics.

Meanwhile, in-memory provides a new paradigm that allows data to be analyzed while in-memory, thus avoiding the expensive data movement to storage. In general, in-memory

can be either in situ [43], [44], [45], [46] or in transit [47], [48], [49], [50], [51], depending on how the simulation and data analytics are coupled. For in situ, the simulation and data analytics are allocated on the same node and share the physical memory. As a result, there is no additional data movement. However, the simulation will be interrupted when the analysis occurs. In contrast, in transit ships data from the simulation memory to a dedicated staging area where the data can be further processed. While this approach involves explicit data movement across nodes, a key advantage is that the simulation can run asynchronously with the data analytics, thus posing low impact to the simulation. Although this paper focuses on the persistent storage, the idea can be broadly applied to in transit, where the interference over the interconnect can be mitigated in a similar fashion.

Data reduction has become increasingly important as the forefront to manage the ever-increasing data volume and velocity. Broadly speaking, data reduction can be either lossless compression or lossy compression. Lossless compression, FPC [52], FPZIP [53], GZIP [54], requires data to be identical after decompression, and is often used in the scenarios where the accuracy is strongly enforced, such as checkpoint and restart. However, the compression ratios achieved are often low [55], greatly limiting its adoption to large applications that can easily reach TBs or PBs. Meanwhile, lossy compression, SZ [56], ZFP [57], ISABELA [58], achieves significantly higher reduction ratios with low overhead by trading accuracy for performance. More broadly, the data decimation that is used by this paper to generate the reduced representation is also a form of lossy compression by simply discarding selected data points. A key drawback of decimation is the substantial loss of information. This work addresses this weakness by providing additional augmentation, if needed. Canopus [27] is a recent work that utilizes decimation to map simulation data to storage hierarchy. Our work differs in that we exploits the HPC interference pattern to provide augmentation on-the-fly.

VI. Conclusion

This paper aims to address the I/O performance variation issue on HPC systems. In contrast to the prior efforts that are mostly done in the storage and file system level, we believe there are opportunities in the application space, and data analytics and storage need to be co-designed to mitigate I/O variation. In particular, we propose the idea that data analytics should adapt to the storage interference by trading accuracy for performance - if there is strong interference experienced, the data analytics can lower the accuracy by retrieving less data from the storage system, thus reducing the performance variation. To achieve this, we design a DFT-based estimation scheme to predict the bandwidth consumption of HPC systems, leveraging the fact that HPC applications follow the repetitive patterns. The degree of augmentation is further determined by an augmentation-bandwidth plot. We evaluate three applications, fusion XGC, astrophysics GenASiS, and computational fluid dynamics Jet, on Chameleon, and the results show that both the mean and variance of the I/O performance are vastly improved, and at the same time the outcome data analytics are acceptable. We envision that this work, which is an application level solution, is orthogonal and complementary to many other storage layer solutions, including the load balancing and other advanced QoS solutions. We plan to further combine our method with existing storage QoS solutions in the future work.

Acknowledgments

The authors wish to acknowledge the support from the US Department of Energy Exascale Computing Project (17-SC-20-SC), Chameleon Cloud, and US National Science Foundation CCF-1718297, CCF-1812861.

References

[1] I. Foster and et al., "Computing just what you need: Online data analysis and reduction at extreme scales," in *European Conference on Parallel Processing (Euro-Par'17)*, Santiago de Compostela, Spain, 2017.

[2] "Titan at Oak Ridge Leadership Computing Facility." [Online]. Available: https://www.olcf.ornl.gov/titan/

[3] "Summit at Oak Ridge Leadership Computing Facility." [Online]. Available: https://www.olcf.ornl.gov/summit/

[4] 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.

[5] "Theta at Argonne Leadership Computing Facility." [Online]. Available: https://www.alcf.anl.gov/alcf-resources/theta

[6] S. Di, Y. Robert, F. Vivien, and F. Cappello, "Toward an optimal online checkpoint solution under a two-level hpc checkpoint model," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 244–259, Jan 2017.

[7] 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.

[8] 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.

[9] 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.

[10] 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, p. 155?164. [Online]. Available: https://doi.org/10.1109/IPDPS.2014.27

[11] 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.

[12] 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, p. 309?320. [Online]. Available: https://doi.org/10.1145/2745844.2745876

[13] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[14] 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.

[15] C. W. Rowley and S. T. Dawson, "Model reduction for flow analysis and control," *Annual Review of Fluid Mechanics*, vol. 49, pp. 387–417, 2017.

[16] 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.

[17] 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.

[18] 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, p. 307?320.

[19] 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, p. 437?450.

[20] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "Crush: Controlled, scalable, decentralized placement of replicated data," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, Nov 2006, pp. 31–31.

[21] "Lustre software release 2.x operations manual," http://doc.lustre.org/lustre_manual.pdf, Accessed: 3-4-2020.

[22] L. Xi and Z. LingFang, "Lime: A framework for lustre global qos management," https://www.eofs.eu/_media/events/lad18/03_li_xi_lad2018_-qos.pdf, Accessed: 3-4-2020.

[23] "Spectrum scale," https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.v5r04.doc/bl1adm_mmlsqos.htm, Accessed: 3-4-2020.

[24] "Orange file system 2.9.7," :http://www.orangefs.org, Accessed: 3-4-2020.

[25] 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.

[26] "Cori scratch at NERSC." [Online]. Available: https://docs.nersc.gov/filesystems/cori-scratch/

[27] 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.

[28] M. Isenburg and J. Snoeyink, "Mesh collapse compression," in *SCG '99*, Miami Beach, Florida, USA, 1999.

[29] P. Wesseling, *An introduction to multigrid methods*, ser. Pure and applied mathematics. John Wiley & Sons Australia, Limited, 1992. [Online]. Available: https://books.google.com/books?id=MznvAAAAMAAJ

[30] 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.

[31] 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.

[32] 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.

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

[34] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. USA: Prentice-Hall, Inc., 1993.

[35] ASCAC Subcommittee, "Top ten exascale research challenges," 2014. [Online]. Available: https://science.energy.gov/ /media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf

[36] 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.

[37] 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

[38] 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.

[39] 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*, Nov 2008, pp. 1–12.

[40] 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.

[41] Y. Gong, B. He, and D. Li, "Finding constant from change: Revisiting network performance aware optimizations on iaas clouds," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 982–993.

[42] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, and R. Ricci, "Taming performance variability," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 409–425. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/maricq

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. Wegener, "Assessing the effects of data compression in simulations using physically motivated metrics," *Scientific Programming*, vol. 22, no. 2, pp. 141–155, 2014.

[56] S. Di and F. Cappello, "Fast Error-Bounded Lossy HPC Data Compression with SZ," in *IPDPS'16*, 2016.

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

[58] 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.