Restructuring Data for Exascale Scientific Productivity

Hasan Abbasi[‡] , Jay Lofstead[*] , Mark Ainsworth[‖‡] , Jong Choi[‡] , Matthew Curry[*] , Scott Klasky[‡§**], Tahsin Kurc[‡‡], Qing Liu[‡] , Jeremy Logan[§] , Carlos Maltzahn[†] , Manish Parashar[¶] ,
Norbert Podhorzski[‡] , Feiyi Wang[‡] , Matthew Wolf[**], Ben Whitney[‖] , C. S. Chang[††], Michael Churchill[††], Stephane Ethier[††]
[*] Sandia National Laboratories, [†] University of California Santa Cruz,
[‡] Oak Ridge National Laboratory, [§] U. Tenn. Knoxville, [¶] Rutgers University,
[‖] Brown University, [**]Georgia Tech, [††]PPPL, [‡‡]Stony Brook

ABSTRACT

Big data and the associated changes in the computing platform are key components for the advances from exascale systems for scientific discovery with computing. Data will increasingly drive the next generation of scientific discoveries, but limitations in the rate of progress for data storage and movement technologies is likely to hamper this revolution. Storage performance and capacity bottlenecks have the potential to greatly limit how much data can be output from exascale simulations, how often this data can be output, and how much knowledge can be derived from the data.

New techniques in storage and I/O hardware is one direction towards addressing these issues. In this paper we present an alternate method for addressing the data deluge, by utilizing domain specific knowledge about the data to manage the organization, content and precision of the output data. The tradeoff between system level quality of service and domain specific quality of information can be used to circumvent the storage bottlenecks without exploding the cost or energy usage of storage systems.

In this paper we present a lossless data partitioning technique that intelligently utilizes hierarchical storage in next-generation leadership computing systems. We show that lower accuracy partitions can be used to service the need for most common use cases, and additional accuracy can be added for more demanding scenarios. The performance benefits are .....

1. INTRODUCTION

Big data and the associated changes in the computing platform are key components for the advances from exascale systems for scientific discovery with computing. Data will increasingly drive the next generation of scientific discoveries, but limitations in the rate of progress for data storage and movement technologies is likely to hamper this revolution. Storage performance and capacity bottlenecks have the potential to greatly limit how much data can be output from exascale simulations, how often this data can be output, and how much knowledge can be derived from the data.

New techniques in storage and I/O hardware is one direction towards addressing these issues. In this paper we present an alternate method for addressing the data deluge, by utilizing domain specific knowledge about the data to manage the organization, content and precision of the output data. The tradeoff between system level quality of service and domain specific quality of information can be used to circumvent the storage bottlenecks without exploding the cost or energy usage of storage systems.

We have undertaken this effort as part of a larger rethinking of the role of storage in high performance computing applications, the SIRIUS project. In this paper we demonstrate the initial efforts towards creating precision aware I/O techniques for different types of large scale data sets. Our experiments show that selecting for output accuracy based on the intended usage for data can yeild substantial performance benefits. Moreover, we show that lossless data partitioning techniques can be utilize in lieu of lossy compression methods to utilize the hierarchical storage structure of next-generation leadership computing platforms. By partitioning data into buckets of varying precision, many usecases can be met with smaller, less accurate buckets. For other use cases where only the highest precision will suffice, the additional cost can be paid as and when needed.

This paper describes our initial foray into hierarchical partitioning techniques to match the structure of storage in future systems. We have developed three sets of techinques for partitioning scientific data from large scale applications, evaluated these techniques on our local cluster to gain deeper understanding of the cost-benefit tradeoffs with respect to computational time, storage time, storage space, access speeds and power consumption, and utilized microbenchmarks to extrapolate our results to future use cases. Our research is driven by a combination of application user requirements and data consumption use cases, such as scientific visualization. To aid in adoption and experimentation, we have integrated our techniques with the ADIOS high performance I/O middleware library and the Visit visualization application.

PDSW15 November 16, 2015, Austin, TX, USA

ACM 978-1-4503-2505-9/13/11...$15.00.
http://dx.doi.org/xxxx.xxx/xxxxxxxxxx

In the next section, Section 2, we present the application use case for our data partitioning approach, and a discussion on the storage hierarchy in Section 3. The three broad techniques we study in this paper are bit-based splitting of double precision floating point data, spatial dimensional sampling of data, and a temporal interpolation based data prediction technique. In Section 4 we provide details on each of these techniques and discuss the various tradeoffs each technique emobdies. Other research efforts and directions are discussed in Section 8. Our evaluation platform presents a two level storage hierarchy with local SSDs and a remote lustre parallel file system. Both storage targets have different performance characteristics, and more details on the evaluation platform are presented in Section 5. With this background we present our results in Section 9 and discussion in Section **??**. This work is the first step in a comprehensive new direction for data management at large scale and we discuss the future of this effort and conclusions in Section 10.

Contributions:

1. XGC1 Byte-split data (3/5)
   (a) decomposition/compression
   (b) auditor
   (c) data challenges preventing arbitrary use
2. XGC1/S3D Data Interpolation (timestep to timestep)

Engineering:

1. ADIOS write transports
2. VisIt auditor reader
3. interfaces

validation/experiments (XGC1 only for now, 1 &1 2 buckets, all other factors):

1. Performance writing (A orig, B 2 buckets)
2. Performance reading (A orig, B 2 buckets, C bucket 1, D, bucket 2)
3. Visualization diff (3-d)
4. numeric evaluation diff (to get past criticism of 3-d hiding errors)
5. derived quantity
6. interpolation validation

target: Lustre, DataSpaces
nodes: 10, 100, 1000 (Sith 35 compute + 5 storage)
data/node: 10M, 1G
vars: 10, 100
per var: 1, 2 buckets (4 if we have time)
timesteps: 2, 4, 8, 16
time between timesteps: 10s, 100s

XGC1:
1d array format
nparticles (N x 8)
fields (nv) (3-5 vars)

Pixie3D/S3D:
3d array: (1k x 1k x 1k)

Use ITER mesh input for XGC1
Viz diff
{reduced, full, diff} images, numerical eval of diff

Responsibilities:

- Hasan - write draft/drive paper
- everyone - experiment runs
- Ivan - DataSpaces help
- Tahsin/Jong - Use real XGC1 data in test harness
- Tahsin/Jong - Big run for byte split example
- Tahsin/Jong - small run for interpolation example
- Hasan - Real S3D run
- Hasan/Norbert - ADIOS writer
- James/Dave - ADIOS reader/viz
- Hasan/Ben - ADIOS interpolation pieces
- Jay/Carlos - Storage hierarchy information

COMPLETED WORK:
1. XGC1 ADIOS write to VisIt read, 3/5, 4/4, 2/6 split (4/4 converts the 4 double bytes to a proper float). Output is written to disk and read is directly. DataSpaces to do still. all scalars are written to both files.
2. Can we do this to Integers too?
3. XGC1 interpolation uses every 4th element only and did 3/5 split
4. Hasan to talk with Jeremy about how to make the interpolation work properly. 5. Doing off the particles for the first pass. Checking to see if they can use the fields instead.
6. Also measure the diff with the viz and the numerical eval for the 3/5 split
7. Make it neutral in terms of tech used to make double blind possible. Add in the rest of the details once accepted

2. MOTIVATION AND APPLICATION USE CASES

3. STORAGE HIERARCHY

This is the storage section

4. PARTITIONING TECHNIQUES

Modern experiments and numerical simulations produce datasets so large that their storage and retrieval strain the capabilities of storage hardware. Slow writing of the data is problematic on shared machines where memory is periodically purged and transferred to new users; slow reading delays analysis of the data after the experiment is over. Compression offers a solution to this problem. A crude but commonly used compression technique is decimation, wherein

only every sth datum of a sequence is stored. In practice, the decimation factor s, also known as the stride, is often chosen to be 10. The remaining data is simply discarded. Clearly, decimation trades a reduction in the size of the dataset for data loss. The loss can be mitigated by generating approximations to the discarded data from the retained data. A simple, natural technique for doing so is linear interpolation. Let $\{u_j\}_{j\in\mathbb{N}}$ be the original dataset. To generate an approximation $\tilde{u}_j$ to a discarded datum u_j, we find stored values u_{ms} and u_{ms+s} that straddle u_j $(ms < j < ms + s)$ and take a weighted average:

$$u_j \approx \tilde{u}_j = (1 - \tfrac{j-ms}{s})u_{ms} + \tfrac{ms+s-j}{s}u_{ms+s}.$$

Observe that decimation paired with interpolation, illustrated in Fig. 1, gives a compression ratio of s and provides a full set of values at all original timesteps. This procedure

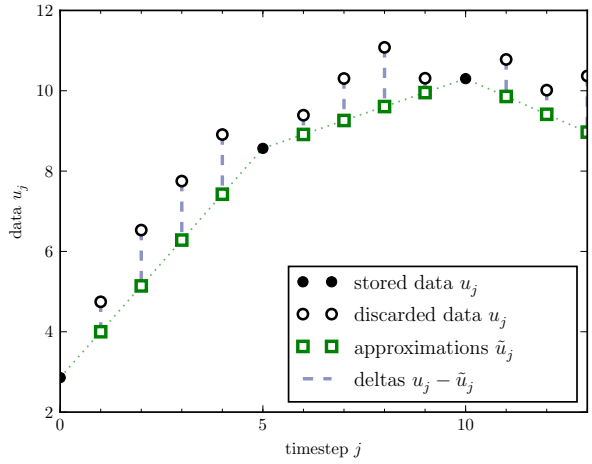

Figure 1: Illustration of decimation procedure for partitioning of data in the case $s = 5$.

produces a good compression ratio, and it is computationally attractive: for a given j only two stored values need to be retrieved in order to reconstruct a value for u_j. However, as noted above, there is a direct tradeoff between the compression ratio and the data loss. Nonetheless, despite these obvious shortcomings, the above approach or variants thereof is in widespread use as a technique for compression large datasets, particularly in the context of computational simulation. In our implementation, the data is partitioned into the *top level data* $\{u_{ms}\}_{m\in\mathbb{N}}$, consisting of the decimated data, and the *bottom level data* $\{u_j - \tilde{u}_j\}_{j\in\mathbb{N}}$, consisting of the differences, or *deltas*, between the original data and the approximated data.

Typically, the deltas resulting from decimation paired with interpolation are simply discarded. The tacit assumption behind this approach is that the approximated dataset $\{\tilde{u}_j\}_{j\in\mathbb{N}}$ is a sufficiently good approximation to the full dataset $\{u_j\}_{j\in\mathbb{N}}$. The issue of the accuracy of this process has received little or no attention, but the following result is proved in [1]:

$$\max_{j\in\mathbb{N}}|u_j - \tilde{u}_j| \le \min(2\mathfrak{M}_0, \tfrac{1}{2}s\mathfrak{M}_1, \tfrac{1}{8}s^2\mathfrak{M}_2) \qquad (1)$$

where $\mathfrak{M}_0 = \max_{j\in\mathbb{N}}|u_j|$, $\mathfrak{M}_1 = \max_{j\in\mathbb{N}}|u_j - u_{j+1}|$, and $\mathfrak{M}_2 = \max_{j\in\mathbb{N}}|u_j - 2u_{j+1} + u_{j+2}|$. These bounds are illustrated for a particular dataset in Fig. 2. Notice that when s is smaller, the accuracy is high, meaning that the top level data produces an acceptable surrogate for the full original dataset. However, for larger strides (or, equally well, for higher compression ratios), the accuracy becomes unacceptable. The bottom level data, containing the deltas, enables lossless reconstruction of the data when the accuracy of the approximations is unacceptable for a particular purpose.

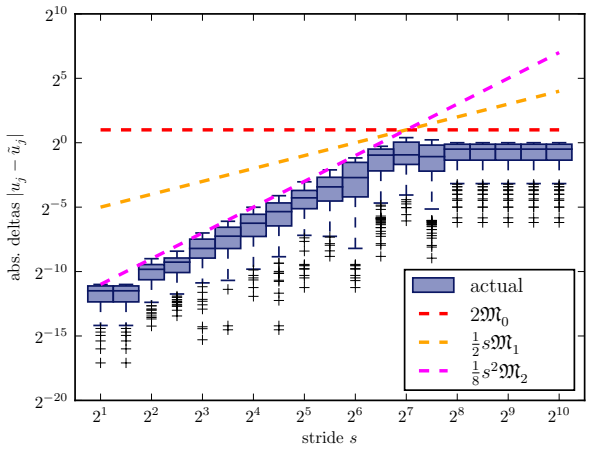

Figure 2: Illustration of the bounds in Eq. (1) for a range of strides in the case $u_j = \sin(2^{-5}j)$ for $j = 0, \ldots, \lfloor 2^6\pi \rfloor$. Observe the transition between different cases of Eq. (1) at $s \approx 2^7$.

One might question whether this partitioning procedure provides any overall compression of the original data. The amount by which a dataset can be compressed is quantified by its Shannon entropy. One advantage of the decimation–interpolation process is that for reasonably smooth data the entropy of the deltas $\{u_j - \tilde{u}_j\}_{j\in\mathbb{N}}$ will be smaller than that of the original data stream $\{u_j\}_{j\in\mathbb{N}}$. Consequently, the deltas are more amenable to compression using an entropy encoder than were the original data. The following result is proved in [1].

THEOREM 1. *The average differential entropy per delta in $\{u_j - \tilde{u}_j\}_{j\in\mathbb{N}}$ is bounded by the minimum of*

(1) $1 + \log_2 \mathfrak{M}_0 + \frac{1}{s-1}\log_2\left[\frac{1}{6}(s+1)(s+2)\right]$

(2) $1 + \log_2 \mathfrak{M}_1$

(3) $1 + \log_2 \mathfrak{M}_2 - \frac{1}{s-1}\log_2 s$

These entropy bounds can be translated to bounds on the expected compression ratio achieved by the combination of decimation, interpolation, and compression of the deltas. See [1] for details and Fig. 3 for an illustration.

5. EVALUATION PLATFORM

6. EVALUATION

7. COST-BENEFIT DISCUSSION

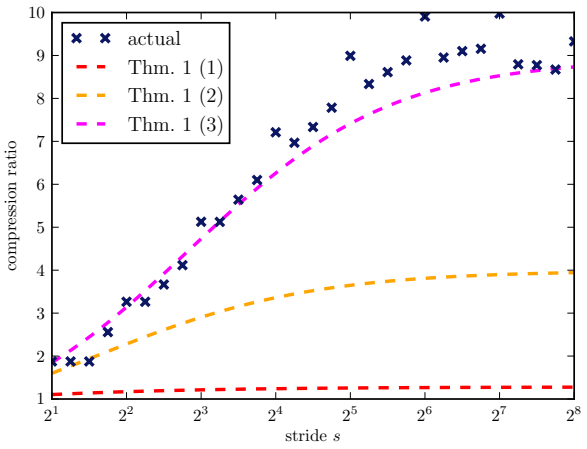

Figure 3: Illustration of the compression ratio bounds derived in [1] from Theorem 1 for a range of strides in the case $u_j = 2^4(3 + \sin(2^{-15}j))$ for $j = 0, \ldots, \lfloor 2^{16}\pi \rfloor$.

8. RELATED WORK

Many projects over the last couple of decades have sought to address some challenging aspect of parallel file system design. The recent rise of "Big Data" applications with different characteristic IO patterns have somewhat complicated the picture. Extreme scale machines will be expected to handle both the traditional simulation-related workloads as well as applications more squarely in the Big Data arena. This will require some adjustments to the underlying system for good performance for both scenarios.

The major previous work is really limited to full file systems rather than the mountain of file system refinements made over the years. A selection of these other file systems and some features that make it relatively unique are described below.

Ceph [9] is a distributed object store and file system. It offers both a POSIX and object interface including features typically found in parallel file systems. Ceph's unique striping approach uses pseudo-random numbers with a known seed eliminating the need for the metadata service to track where each stripe in a parallel file is placed. PVFS [3] offers optimizations to reduce metadata server load, such as a single process opening a file and sharing the handle. It has been commercialized in recent years as OrangeFS. Lustre [2] has become the de facto standard on most major clusters offering scalable performance and fine-grained end-user and programmatic control over how data is placed in the storage system. GPFS [7] offers a hands-off approach for providing good performance for scaling parallel IO tasks and is used extensively by its owner, IBM. Panasas [5] seeks to offer a dynamically adaptive striping system that detects the need for additional stripes for performance and adjusts the file layout as necessary.

Other file systems, like GoogleFS [4] and HDFS [8], address distributed rather than parallel computing and cannot be compared directly. The primary difference between distributed and parallel file systems is the ability of the file system to store and retrieve data simultaneously from multiple clients, in parallel, and treat the resulting collection of pieces as a single object. Distributed file systems rely on a single client creating a file, but distributing the set of files across a wide array of storage devices. The other, popular distributed file system of note is NFS [6] that has been used for decades for enterprise file systems. These other file systems are mainly of interest in the context of the ACG features of FFSIO and will be discussed more in Section **??**.

9. EVALUATION

10. FUTURE WORK AND CONCLUSIONS

11. ACKNOWLEDGEMENTS

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

12. REFERENCES

[1] M. Ainsworth, B. Whitney, and S. Klasky. A practical approach to lossless compression of scientific data: Analysis and applications. In preparation.

[2] P. J. Braam. The lustre storage architecture. Cluster File Systems Inc. Architecture, design, and manual for Lustre, Nov. 2002. http://www.lustre.org/docs/lustre.pdf.

[3] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, Oct. 2000. USENIX Association.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proceedings of the NineteenthACM Symposium on Operating Systems Principles*, pages 96–108, Bolton Landing, NY, Oct. 2003. ACM Press.

[5] Object-based storage architecture: Defining a new generation of storage systems built on distributed, intelligent storage devices. Panasas Inc. white paper, version 1.0, Oct. 2003. http://www.panasas.com/docs/.

[6] B. Powlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel, and D. Hitz. NFS version 3: Design and implementations. pages 137–152, 1994.

[7] F. Schmuck and R. Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proceedings of the USENIX FAST '02Conference on File and Storage Technologies*, pages 231–244, Monterey, CA, Jan. 2002. USENIX Association.

[8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[9] 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 2006Symposium on Operating Systems Design and Implementation*, pages 307–320. University of California, Santa Cruz, 2006.