Exascale Storage Systems the Sirius way [4 pages]

Scott A. Klasky136, Hasan Abbasi1, Mark Ainsworth51, Jong Choi1, Matthew Curry2, Qing Liu1, Jay Lostead2, Carlos Malzahn4, Manish Parashar4, Norbert Podhorszki1, Feyi Wang1

1Oak Ridge National Laboratory, 2Sandia National Laboratory, 3U. Tenn. Knoxville, 4Rutgers University, 5Brown University, 6Georgia Institute of Technology

*Abstract*— As the exascale age of computing emerges data become one of the critical factors in how and where we do computing. Popular approaches in the past for I/O and storage libraries become increasingly bottlenecked by their past assumptions on data movement, re-arrangement, and storage. New technologies of today such as “bust buffers” will help address dome of the short-term performance problems, but the underlying storage and I/O infrastructure needed to address the exascale age must be re-examined. Our approach is to try to make an important distinction between the data, metadata, and knowledge contained in the vast amount of information generated, and understand it’s utility as the data ages in time so that we can formulate a new approach to the exascale Storage System and I/O (SSIO) which involves allowing users to place their application knowledge into the system to better manage, store, and access the knowledge contained in the voluminous data.

# Introduction

The purpose of computing is insight, not numbers [2], a remark Richard Hamming made over 50 years ago at the dawning of the age of large scale scientific computing. The central objective of our next-generation storage and I/O system (Sirius) is to be able to minimize the time to insight with respect to scientific computing, and this means that we need new research into managing, storing, and retrieving the large volumes of data that are produced by simulations and analyzed for months afterwards. To illustrate our point of the necessary R&D needed to eliminate much of the current and future problems we will first give an illustrative example, and from this example talk about the different areas we will concentrate in. Later we will discuss a new approach to refactoring of data which can help eliminate much of the current and future bottlenecks in knowledge discovery but this generates new challenges from both the applied mathematics and computer science computers. Finally, we will discuss this new approach of using an auditor to help refactor the data and the additional meta challenges this creates for exascale storage systems.

## Illustrative Example

The motivating use case is a series of simulations of the ITER fusion experiments using the XGC1 application. XGC1 is one of the largest applications used at the Department of Energy's (DOE's) Leadership Class Facilities, with an allocation of over 300M hours. These simulations were able to produce 100 PB of data over ten days of total runtime on the Titan system at the Oak Ridge Leadership Computing Facility (OLCF), and required a team of experts including the user group and I/O and storage personnel to help ensure that the maximum amount of information would be saved. Due to physical resource limitations, the simulation was immediately restricted to write about 10 PB. However, when the time to write and read, together with the financial cost to archive this volume of data was fully explored, the data size goal was further reduced to only 5 PB over the ten-day run.

This forced a careful scrutiny of the large amount of data to be generated and restriction of the output to only the most important pieces. Further, this reduced data set had to be divided into two categories: first, the data that would be accessed while the simulation was running or shortly thereafter, and second, the data that would be needed after the whole campaign was finished. The former category of data would be stored on disk, and the latter would be archived. Additionally, the entire team needed to figure out how to efficiently retrieve these different categories of data for post-processing, analysis and visualization.

To accomplish this, the application and storage teams developed new application specific data reduction techniques and added them to the ADIOS I/O middleware layer. A rudimentary discovery system was also created to track what data was on disk and what was on the tape archive. Initially this solution was sufficient, but eventually a new problem was introduced: Although the team had reserved resources for the run, the entire file system was shared resulting in very high I/O variability. Addressing this variability, caused by contention on shared resources, is an ongoing concern.

The problems faced by this user workflow motivate our research here. The use case demonstrated the three tiers of solutions that are required, application level knowledge of data and how it will be used, middleware management of data and resources, and storage system level scheduling of resources. It also provided us with new insights: First, the users and developers of the application data can, without a great loss of encoded knowledge, reduce the size of data that needs to be stored. Second, data storage must be managed with input from the users to correctly decide the target of storage operations. Third, shared resources must be managed in a way that provides sufficient performance to all applications without encumbering the applications with high variability in performance. These insights drive the following sections of our work.

## Two guiding principles of Sirius

Our challenge in the exascale storage and I/O community is to overcome the challenges described above for both the application scientist trying to write data, and those trying to read data, in a coherent system such that the SSIO system can fairly share the resources among the users but help better enable exascale science by prioritizing and understanding what is being done with more application level knowledge which is then handed down from the application to the storage layer. We are ultimately guiding by our two basic principles:

Principle 1**:** **A knowledge-centric system design** that allows user knowledge to define data policies. Today SSIO layers are written in a stove-pipe fashion, and quite often do not allow optimizations to take place. We are re-designing the layers in a highly integrated fashion where users place their intentions into the system and actions will statically and dynamically take place to optimize for both the system and for individual requests.

Principle 2: **Predictable performance and quality of data in the SSIO layers** needs to be established so science can be done on the exascale systems in a more efficient manner. Without predictable performance, not only can the runs be slowed down because of shared resource contention, but also it affects key science decisions, e.g., how much data reduction should be performed.

We have already seen in ADIOS that we can alleviate the need for the “magic'” and “tricks'” that are currently required to optimize application I/O performance on today's file systems by placing extra annotations in the ADIOS metadata to better understand the underlying storage system. In SIRIUS we are further accomplishing this by providing a systematic approach for describing intentions and other knowledge from the *user*, as well as allow performance estimations and guarantees from the underlying storage.

By capturing user intentions and acting upon them in the middleware, we free the user from polluting application code with system specific optimizations which may provide better performance in the short term, but are detrimental to the long-term maintainability of applications, and are more sensitive to changes in the system configuration. We have successfully employed this strategy of separation of concerns in ADIOS, and believe that it will become increasingly important because saving all data may not be possible, and users want the ability to describe and prioritize different chunks of data.

These techniques will be further integrated in combing ADIOS with RADOS-Cepth, which is a distributed object store and file system. Cepth offers both a POSIX and object interface including features typically found in parallel file systems.

# Data Refactoring

The classical workflow where the entire dataset is written to storage for later analysis will no longer be viable in exascale simply because the amount of generated data will be too large due to capacity and performance reasons.

In the future, it will be vital to take advantage of *a priori* information when (1) writing and reading from the storage system to gain higher performance and predictability and prioritize the most useful data for end users so that I/O can be finished in the time available, and (2) performing *in situ* operations and analysis before storing the information.

In the fusion use case described earlier which shares commonalities with many other DOE applications, *a priori* information can indeed be provided by application scientists regarding which data should be sent to the storage system, which we refer to as SIRIUS, so that minimally, the most science relevant data can be available for subsequent analysis. This allows the science to be done even when the storage is busy (we believe in a shared multi-user environment). It occurs often and causes high performance variability. This capability of prioritizing data is accomplished by data refactoring, which includes data re-organization and reductions. There are many methodologies to refactor data and the best choice will generally be application dependent. However, our observation is that, once the choice is settled for an application, it will not typically change from run to run.

To refactor data effectively and efficiently, it is important to understand the critical point when the time and resources required to identify and perform the “best methods” outweighs the intended gains. Another critical research question concerns the quantification and control of the loss of information resulting from refactoring the data and using a reduced dataset. A basic issue allied to refactoring is to understand how much information is actually present in a dataset and whether a refactoring based on a reduced order representation might prove effective. It is useful to classify scientific data into two basic categories: regular data (in the sense that it satisfies a known or relatively simple model), or irregular data (in which information content follows no obvious or explicit model). Although scientific data generally contains random components (due to finite precision and measurement and calibration effects), useful scientific data is never purely random. Broadly speaking, the path from data to knowledge consists of extracting underlying models or patterns from the datasets and interpreting the resulting models.

Ideally, scientists would like to perform the entire analysis stage *in situ* to extract the relevant information effectively circumventing the large data issue completely. The catch, of course, is that thisis unlikely to be possible since, by their nature, large scale simulations aimto discover new information often hidden in the form of higher ordereffects amongst the data deluge. In particular, this means if data thinning ortruncation is applied haphazardly, the sought data may be eliminated.Theentire data set cannot be stored in an easily accessible source due to sheer size. Yet, the data cannotbe reduced prior to archiving without risking losing the desired information.Viewed in this way, the problem would appearintractable. However, the above discussion fails to recognize that much of thedata is redundant in an information theoretic sense. That is to say, theamount of information contained in the dataset is often significantly less thanthe amount of data. The difficulty stems from the fact that one does not knowabout this redundancy in advance *without the benefit of a priori knowledge.* This knowledge can often be provided by the user or can be acquiredthrough experience of dealing with differing runs of the same code.

Deep application knowledge means one can sometimes achieve dramatically superior data reduction compared with what one might achieve otherwise. However, even in the absence of such high level knowledge, we must equip SIRIUS with generic data reduction and re-organization techniques. For instance, certain basic data semantics information is needed and must be supported by the overall infrastructure. Currently we are studying three generic, and one application specific refactoring methods: 1) Precision based, 2) Frequency-based, 3) Linear Auditing, and 4) Application aware histogramming, which we describe below.

**Precision based refactoring** is where the most significant bytes of the data are all grouped together from each object. This data generally will have a higher utility than the data with the least significant bytes. The data needs to be re-arranged which involves potentially intensive memory operations and needs to be done as much in situ as possible without involving the slow storage systems and inter-processor communications. At most this requires two copies of an individual dataset in memory, and we are investigating techniques to allocate and deallocate this memory if the user will specify that the data will be overwritten after it is written to the storage system. This is often the case for many of the quantities written from the simulation, but there are many cases where, for example, we want to write all of the particles from a Particle-In-Cell (PIC) simulation. Since the particles will be used later in the calculation, we need to duplicate the storage. Our observation with working with the XGC1, GTC, and PiconGPU simulations is that we can temporary increase the storage of the particles, and then release them since the temporary arrays used in the calculations are often freed when a PIC iteration is finished. Another challenge which we are facing is that when we read back the data there is a serious potential cost in reconstructing the data. In our example we can imagine that the most significant bits are written to the parallel storage system. We see that the cost savings to store

**Frequency-based refactoring** Another common approach to classifying the importance of data is to re-organize data according to it frequency. This is commonly used in streaming data and data reduction techniques such as those used JPEG-2000. These mechanisms support spatial random access or region of interest access at varying degrees of granularity. It demonstrates the possibility of storing the same data using different quality. In relation to this project, this capability allows us to place the lowest frequency chunks in the fastest storage and the highest frequency chunks on either the slowest storage tiers, or if writing out the data sizes are prohibitively costly, not even write out these pieces. In order to fully take advantage of frequency based re-organization, the team has devised a scheme where the data pre-conditioner first sorts the data in bins with fixed length, and then wavelets and spline fits are used to re-organize the values. This re-organization effectively produces a smoother data allowing for much better reduction later.

**Linear Auditor** is another technique which uses a delta compression algorithm in time to perform lossless (and lossy) compression of data. This technique has its basis in information theory where we understand that smooth data has low entropy, and noisy data has high entropy and the thought is to separate the low entropy part of the data from the high entropy part. We can have about ½ of a page on this linear auditor/. Mark can you put in some preliminary result for this as well

**Application Aware Auditor** contains a class of techniques which have more domain knowledge than the generic techniques described above. One algorithm we are looking at is combining continuum based physics knowledge into Particle-In-Cell particle data, to create a two-dimensional histogram of the data (in the XGC1 case we will histogram in the velocity dimensions of the particles, and then overlay a Maxwelliam distrubtion to this, and see the deviaions and keep the differences of the deviations which will greatly reduce the number of particles. We have been working with C. S. Chang group at PPPL to achieve this historgram approach for the XGC1 code, and have seen that we can reduce the data by 100X while retaining over 90% accuracy for all of known analytics we have worked on.

# Computer science challenges in refactoring

Data that comes from the simulation will be greatly modified in order to efficiently separate out the different levels of imHasan’s section

# metadata challenges

Jay Lofstead should fill out this part.

# Conclusion

None yet

Acknowledgment

Sirius project….

References

1. http://www.ieee.org/publications\_standards/publications/authors/authors\_journals.html
2. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions*,” Phil. Trans. Roy. Soc.* London, vol. A247, pp. 529-551, Apr. 1955.
3. J. Clerk Maxwell, *A Treatise on Electricity and Magnetism,* 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp. 68-73.
4. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in *Magnetism,* vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
5. T. L. Gilbert, Formulation, Foundations and Applications of the Phenomenological Theory of Ferromagnetism, Ph.D. dissertation, Illinois Inst. Tech., Chicago, IL, 1956, unpublished.
6. D. P. Arnold, “Review of superconducting radio-frequency cavities,” submitted for publication.
7. L. Rossi, “Conductor choices for upgrades of CERN magnets,” *IEEE Trans. Appl. Supercond.* **23** submitted for publication.
8. S. O. Demokritov and V. E. Demidov, “Micro-Brillouin light scattering spectroscopy of magnetic nanostructures,” *IEEE Trans. Magn.,* to be published.
9. C. J. Kaufman, Rocky Mountain Research Laboratories, Boulder, CO, private communication, 2004.
10. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” *IEEE Transl. J. Magn. Jpn.,* vol. 2, pp. 740-741, August 1987 [*Dig. 9th Annual Conf. Magn. Jpn.,* p. 301, 1982].