Exascale Storage Systems the Sirius way [4 pages]

Scott A. Klasky136, Hasan Abbasi1, Mark Ainsworth51, Jong Choi1, Matthew Curry2, Qing Liu1, Jay Lostead2, Carlos Malzahn7, Manish Parashar4, Norbert Podhorszki1, Feyi Wang1

1Oak Ridge National Laboratory, 2Sandia National Laboratories, 3U. Tenn. Knoxville, 4Rutgers University,

5Brown University, 6Georgia Institute of Technology, 7University of California, Santa Cruz

*Abstract*— As the exascale computing age emerges, data related issues are becoming critical factors that determine how and where we do computing. Popular approaches used by traditional I/O solutions and storage libraries become increasingly bottlenecked due to their assumptions about data movement, re-organization, and storage. While, new technologies, such as “burst buffers”, can help address some of the short-term performance issues, it is essential that we reexamine the underlying storage and I/O infrastructure to effectively support requirements and challenges at exascale and beyond. Our approach is to make an important distinction between the data, metadata, and the knowledge contained therein and to understand it’s utility as the data ages in time. In this paper we present a new approach to exascale Storage Systems and I/O (SSIO), which is based on allowing users to inject application knowledge into the system and leveraging this knowledge to better manage, store, and access large data volumes so as to minimize the time to scientific insights. Central to our approach is the distinction between the data, metadata, and the knowledge contained therein, the understanding of the utility of data as it ages in time.

# Introduction

The purpose of computing is insight, not numbers [6], 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 minimize the time to insight with respect to scientific computing at extreme scales. This means that we need new research into managing, storing, and retrieving the large data volumes produced by simulations, and analyzed for months afterwards.

In this section, we first present an example to illustrate these challenges and to motivate the research components of the SIRIUS project. We then describe the key applied mathematics and computer science research components of SIRIUS. These include a new approach to refactoring data that can eliminate much of the current and future bottlenecks in knowledge discovery called an *auditor*.However, this generates new challenges for both applied mathematics and computer science computers. The auditor approach helps refactor the data as well as the systems solutions required to manage the data lifecycles at extreme scales.

## 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 scheduled 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 total.

This forced a careful scrutiny of the 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: the data that would be accessed while the simulation was running or shortly thereafter and 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 tape. Initially this solution was sufficient, but eventually a new problem was discovered. Although the team had reserved compute 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. Earlier efforts [11] are insufficient due to the immense data volumes, limited time available, and the number of other users of the shared storage array.

The problems faced by this user workflow motivate our research here. The use case demonstrated three tiers of solutions 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 SIRIUS project and its research components.

## Guiding principles of Sirius

The exascale storage and I/O community must overcome the challenges described above for both the application scientist trying to write data and those trying to read data. This must be done such that the SSIO system can fairly share the resources among the users while helping enable exascale science by prioritizing and understanding the application level data requirements. The SIRIUS project is ultimately guided by two basic principles:

Principle 1**:** **A knowledge-centric system design** that allows user knowledge to define data policies. Since not all data has equal value for scientific insights, managing data based on features or entire data sets based on the number of outputs since a high fidelity output can offer acceptable data quality with increased data quantity. Further, using a single storage layer for an output limits potential performance. By moving away from the layer isolated IO model to using an integrated storage approach, we can incorporate optimizations existing SSIO layers written in a stovepipe fashion do not allow. We are re-designing the layers in a highly integrated fashion where users express their intentions into the system and actions will statically and dynamically optimize for both the system and for individual requests.

An implied idea within a knowledge-centric system design is making sure the right data is in the right place at the right time. We are investigating not just proactive data placement, but also data migration to address data use needs.

Principle 2: **Predictable performance and data quality in the SSIO layers** needs to be established to maximize the information (rather than raw data) generated on the exascale systems. Without predictable performance, runs may be slowed down because of shared resource contention, which can affect key science decisions, e.g., how much data reduction should be performed. To have some confidence that sufficient output time is available, a conservative estimate, rather than nearly accurate, can be used.

ADIOS can alleviate the need for user-provided “magic” and “tricks” to optimize application I/O performance on today's file systems. In SIRIUS, we are extending this by providing a systematic autonomic approach for combining intentions and other knowledge from the *user* with 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. We have successfully employed this separation of concerns in ADIOS and believe 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-Ceph, which is a distributed object store and file system. Ceph 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 at exascale simply because the amount of generated data will be too large due to capacity and performance limitations.

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.

The fusion use case described earlier shares commonalities with many other DOE applications. *A priori* information can be provided by application scientists regarding identifying which data should be persisted in the storage system (e.g., SIRIUS) so that minimally, the most science relevant data can be available for subsequent analysis. This allows science goals to be accomplished even when the storage is busy servicing other users. Performance impacts from simultaneous storage system users occur often and causes high performance variability. The data refactoring technique used is partially driven from the data priority and also by storage system performance soft guarantees available. Possible refactoring approaches includes data re-organization and reductions. There are many data refactoring techniques and the best choice will generally be application dependent. As the prioritized data volume changes over the simulation lifetime and the performance available in the storage system, the refactoring technique may change multiple times and even as often as for every output operation.

One research challenge for effectively and efficiently refactoring data is understanding when the time and resources required to identify and execute the “best methods” exceeds the gains achieved. Another critical research question concerns quantifying and controlling information loss from refactoring data and using a reduced dataset. A basic issue associated with refactoring is understanding 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 *regular* and *irregular* data. *Regular* data satisfies a known or relatively simple model and *irregular* data 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 *in situ* effectively circumventing the data movement and storage issue completely. Even if it were possible to address reproducibility or data publication requirements while doing purely in situ analysis, it does not address possible scientific discoveries. Consider that 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 higher order effects may be eliminated.Theentire data set cannot be stored in easily accessible storage due to its sheer size. Yet, the data cannotbe reduced prior to archiving without risking losing desired information.Viewed in this way, the problem would appearintractable. However, much of thedata is redundant in an information theoretic sense. That is, theamount of information contained in the dataset is often significantly less thanthe amount of data. The difficulty stems from not knowing about this redundancy in advance *without the benefit of a priori knowledge.* We seek to bridge this gap by leveraging experience of dealing with differing runs of the same code. While past data access patterns do not provide a general guide for future data access, on a per code basis, a user can indicate what runs should be used for training for optimizing near future large scale runs.

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, SIRIUS must offer 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. Each of these is described below.

**Precision based refactoring** groups together the most significant data bytes from each object. This data generally has a higher “utility” than data with the least significant bytes. The data needs to be re-arranged potentially involving memory intensive operations and needs to be done *in situ* as much as possible specifically avoiding the slow storage systems and inter-processor communications network. At most, this requires two copies of an individual dataset in memory. 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 particles storage and then release them since the temporary arrays used in the calculations are often released when a PIC iteration is finished. Another challenge is that when data is read, there is a serious potential cost in data reconstruction. 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** re-organizes data according to how often it appears. 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 levels. 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. To fully take advantage of frequency-based re-organization, 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 smoother data allowing for much better reduction later.

**Linear Auditing** uses a delta compression algorithm in time to perform lossless or lossy data compression. This technique has its basis in information theory where we understand that smooth data has low entropy and noisy data has high entropy. 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 Histogramming Auditing** is a class of techniques with more domain knowledge than the generic techniques described above. One algorithm combines continuum based physics knowledge into Particle-In-Cell particle data to create a two-dimensional data histogram. In the XGC1 case, we will histogram in the particle velocity dimensions and then overlay a Maxwellian distribution to this to see the deviations and keep the differences of the deviations greatly reducing the number of particles. We have been working with C. S. Chang group at PPPL to achieve this histogram approach for the XGC1 code and have seen that we can reduce the data by 100X while retaining over 90% accuracy for all of the analytics we have worked on.

# **Refactoring Challenges**

Refactoring large volumes of data is a challenging computing problem with three broad challenges.

1. Large volumes of data need to be refactored at speeds while utilizing resources in a manner that does not constrain the performance of data generator (the simulation or experimental data acquisition service).
2. The refactored data has to be created in a manner consistent with the access patterns of the data consumer. This requires predicting the common access patterns for data sets, identifying the most likely patterns and then correctly selecting the appropriate refactoring code path.
3. The abstraction through which data is viewed by the user needs to be consistent in the presence of changing refactoring techniques (in order to minimize the cognitive burden on the user), while providing sufficient transparency to enable both user and system to optimize metrics such as data layout, data acccuracy and predictable performance.

We will describe these broad areas in detail and elaborate our approach for addressing these key challenges next.

## Resource Utilization

The generation of large volumes of data, such as those described earlier, is a costly and resource intensive apparatus. Additional work required by data refactoring can only be justified if the benefits outweigh the costs, and if the overall impact on the performance of the data pipeline is positive. Thus, any additional task to refactor, reorganize or compress data will need to account positively for the consumption of resources. In particular, there are three resources that are critical to scientific big data pipelines, on board system memory (RAM), additional time on the processor (CPU), and bandwidth consumed for data movement (network and disk).

Lets consider memory as the first constraint. Data refactoring requires maintaining an original baseline state in memory, while generating the pieces to represent the refactored state.

In the precision based refactoring technique, for example, the original data is kept in memory while tiered precision arrays are generated. At a minimum this requires 2 the size of the required memory. In some, very memory constrained, use cases this additional requirement might render the refactoring an unviable approach. We are currently researching methods to refactor data streams by utilizing temporary windows over a stream of data. This windowed approach will limit the additional memory overhead, but will incur a performance penalty when outputting data. Going forward we will evaluate the performance imapct of this approach, as well as study the optimal window sizes for the different refactoring techniques we have described earlier.

Similarly, for CPU and bandwidth consumption, the consumed resources will produce a negative impact on the performance of the application. Some of this can be mitigated by utilizing asynchronous methods to compute and move the refactored data. This will require careful management of when the refactoring computation is called, and when data is moved, to minimize the overhead on the application due to resource contention. Here we will extend our past work on contention avoidance for data movement and resource sharing .

## Refactoring Selection

## Linked Data Abstraction

# 

# Managing Data Lifecycle

SIRIUS aims to manage the overall data lifecycle, including data generation (e.g., from a simulation) or acquisition (e.g., in the case of experimental and observational data), optimized data placement, runtime data management including migration, reorganization and reduction, data consumption for knowledge discovery, archiving for long-term storage, and purging data from the system to optimize system operation. Key research questions addressed by SIRIUS include: (1) How to initially place data so that it can be discovered and consumed efficiently? (2) How can the placement and migration of data across a multi-tiered storage hierarchy be optimized at runtime both to address application needs and system management concerns? (3) How can knowledge about the application used to better prepare the data for consumption? (4) When and how do we make the decision to purge data?

**Data placement and movement:** When an application outputs or accesses data, the storage and middleware layers need to decide what data is placed where in the multi-level storage system. This placement decision can have a significant impact on data management throughout the lifetime of the data. For example, our past work on data staging [12,13] on HPC systems with multi-level memory structures has shown that different output techniques targeting different layers of the storage hierarchy can have a significant impact on the overhead observed by the application for I/O operations. A key requirement is application-driven runtime mechanisms for dynamically managing data placement across the layers of the distributed storage hierarchy throughout the data lifecycle, coordinating data movement and data sharing between the components of the application workflow, with the overarching goal of maximizing the relative utility to the application as well as the system while reducing access costs. As noted before, the complexities of heterogeneous multi-level storage structures requires adaptive placement policies are required to be implemented to optimally utilize storage resources vertically (across deep memory hierarchies) and horizontally (across nodes within a memory level) while accommodating dynamic application requirements and transient system states. Our approach is to increase knowledge about the data and its use within the application and leverage this knowledge to drive data placement and overall management.

**Soliciting application hints:** As described in the previous section, one of the main components of our proposed middleware and storage system is the ability to reorganize, refactor, and reduce data as it is generated and to reorganize and possibly regenerate the data as it is accessed. We carry this principle into the placement and movement of data by allowing applications to define hints and policies that guide what data is placed where. We will explore the use of application hints in two distinct areas. First, we will study the challenges and trade-offs of either augmenting the I/O interface with hints or allowing the addition of an external specification that defines the use case. Our experience with developing modern I/O interfaces has shown that both techniques have value [12,13] and we will investigate the set of hints that are embedded in the application code vs. those that are described within a non-compiled specification. Second, we will study how hints can guide data placement as data is handed off from application to storage (during a write) and from storage to application (during the read). In both cases we will study what minimal set of annotations and hints can allow the storage system to minimize data movement and optimize the resources consumed by I/O.

# metadata challenges

The challenge when applying refactoring techniques, particularly application aware techniques, is how to incorporate sufficient knowledge in the storage system such that an arbitrary future client has sufficient information to recreate the desired information. Additionally, by spreading data across multiple different kinds of storage media that typically have independent namespaces, locating any particular data will be challenging.

SIRIUS will provide sufficient built-in and extensible metadata services to support efficient data access. First, detailed data metadata, such as array dimensions and other similar data, must be visible within the metadata for any effective data selection. Second, each data chunk stored within the storage hierarchy must have some way to address it. As data utility forces data migration within SIRIUS, the metadata must either dynamically track data as it moves or offer a search feature to discover where data currently resides. Which approach is superior and under what conditions still needs to be determined.

Non-user visible metadata must also be maintained. For example, tracking the I/O access time variability for different storage tiers under different conditions should be gathered for statistically informing soft performance guarantees. This and other system-related information must be collected and organized to inform system operation. One major challenge beyond just the collection and synthesizing this data is to incorporate user-level instructions for what data to use when calculating soft performance guarantees. Similar interactions between the user level and system operation statistics will have to be identified and investigated.

# Conclusion

Extreme scale application workflows, such as the XGC1 fusion workflow described in this paper, generate very large amounts of data that must be processed and analyzed to realize potential scientific insights. Managing, storing and retrieving these large data volumes become critical challenges. The central objective of the SIRIUS project is to address these challenges and minimize the *time to insight* for scientific workflows at extreme scales by enabling predictable performance across the storage system. We achieve this by understanding how the storage system can interact with the middleware so that the data with the highest utility is kept on the fastest layers of the storage system for as long as specified. This interaction is combined with user provided priorities to achieve the best data utility for scientific insights in the least amount of time. This paper presented and overview of the SIRIUS project and its primary research components.

Acknowledgment

Sirius project….

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.

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].
11. Jay Lofstead, Fang Zheng, Qing Liu, Scott Klasky, Ron Oldfield, Todd Kordenbrock, Karsten Schwan, Matthew Wolf. "Managing Variability in the IO Performance of Petascale Storage Systems". In Proceedings of SC 10. New Orleans, LA. November 2010.
12. Sun, Qian, Fan Zhang, Tong Jin, Hoang Bui, Melissa Romanus, Hongfeng Yu, Hemanth Kolla, Jacqueline Chen and Manish Parashar. “Adaptive Data Placement For Staging-based Coupled Scientific Workflows.” *Proceedings of the 28th IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC 15),* Austin, TX, USA, November 2015.
13. Jin, Tong, Fan Zhang, Qian Sun, Hoang Bui, Melissa Romanus, Norbert Podhorszki, Scott Klasky, Hermanth Kolla, Jacqueline Chen, Robert Hager, C. S. Chang and Manish Parashar. “Exploring Data Staging Across Deep Memory Hierarchies for Coupled Data Intensive Simulation Workflows.” *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium*, Hyderabad, India, May 2015.