Data Management Challenges of Exascale Scientific Simulations: A Case Study with the Gyrokinetic Toroidal Code and ADIOS

†Lipeng Wan[1], Kshitij V. Mehta[1], Scott A. Klasky*[1], Matthew Wolf[1], H. Y. Wang[2], W. H. Wang[2], J. C. Li[2], and Zhihong Lin[2]

[1]Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA.
[2]Department of Physics and Astronomy, University of California Irvine, USA.

*Presenting author: klasky@ornl.gov
†Corresponding author: wanl@ornl.gov

Abstract

The push towards exascale computing and the recent introduction of multi-petascale supercomputers have enabled science applications to run complex simulations. However, the gap between compute and I/O has grown wider, even as applications seek to generate and persist increasing amounts of data. Optimizing I/O is challenging and remains a bottleneck at scale. In this paper, we present initial I/O performance results of running Gyrokinetic Toroidal Code (GTC) on Summit, a 200 Petaflop system at Oak Ridge National Laboratory. To manage the complex data in GTC, we use ADIOS, an I/O and data management middleware that provides a rich set of APIs to manage and interact with scientific data. We discuss optimizations performed to obtain improvements in I/O performance and identify a set of challenges that will drive the design and development of next generation data management libraries.

Keywords: Computational, GTC, ADIOS, I/O, Scientific Data, Fusion Simulation

Introduction

With the recent push towards exascale computing, the arrival of systems such as Summit[12], Sierra[22], and Sunway TaihuLight[21] has broken the 100 Petaflop peak performance barrier. This has paved the way for science applications to run more complex simulations. Supercomputer architects are putting more focus on making systems more power efficient, in addition to increasing the memory per core on a node. However, the evolution of I/O technologies has not kept up with other architectural innovations, and the gap between compute capability and I/O bandwidth continues to grow wider.

Science applications that look towards exploiting the compute capabilities of modern systems tend to generate increasing amounts of data. It is common for applications running at full scale to write half the size of system memory for checkpointing. For example, the Chimera molecular modeling and simulation system [15] creates 160 TB checkpoints on the Titan supercomputer at Oak Ridge National Laboratory. On Summit, which has 512 GB of main memory per node, checkpoint data would be close to a Petabyte in size.

The Gyrokinetic Toroidal Code (GTC)[1] is a well-known particle-in-cell application that simulates the movement of magnetic particles in a confined fusion plasma. It generates a variety of data that differ in volume and velocity. Checkpoint and particle data generated over the course of a running simulation are projected to be in the range of a few Petabytes. Management of this data is a daunting task; both the file system capacity and the available I/O bandwidth are limiting factors. The GPFS parallel file system on Summit provides a theoretical peak bandwidth of 2.2 TB/s. Writing a Petabyte of data at near peak bandwidth would take over 500 seconds. The use of non-volatile memory, such as the local NVME chips

on Summit nodes, can help in reducing the time to write bursty, high volume data, but the efficient utilization of NVME is not straight-forward. In combination with high volume and high frequency particle data, I/O can easily become a bottleneck. In addition, projects on Summit have a quota of only 100 TB on the parallel file system, which means the application cannot store all the data it generates to persistent storage. Thus, applications need to make smart decisions about how to write data and what data must be preserved.

ADIOS (Adaptable I/O System) [20] is a well-known I/O framework that is used across many science domains for optimizing I/O and data management. ADIOS provides a self-describing data format and a log-based data organization. As part of its data model, ADIOS generates metadata to preserve the logical ordering of a file. It provides a rich set of APIs to tune various parameters related to I/O to obtain optimal performance. As part of this work, the I/O component of GTC has been adapted to utilize ADIOS. ADIOS provides optimized ways to store data which are comprised of physics variables recorded over thousands of timesteps. In this paper, we present preliminary results of using GTC with ADIOS on Summit. We discuss the optimizations performed in ADIOS that lead to performance improvements in GTC. We highlight the challenges that are faced with increasing data volumes and metadata overhead, and discuss the design of a new metadata format in ADIOS for exascale data. The findings of our study provide valuable insight into the I/O characteristics of scientific simulation codes and help us estimate the areas where bottlenecks can occur when applications are run at exascale. Moreover, we also investigate and evaluate some potential approaches to improving the efficiency of scientific data management and discuss the future research directions.

Background

Summit

Summit is an IBM system located at the Oak Ridge Leadership Computing Facility [12]. With a theoretical peak double-precision performance of approximately 200 PF, it is currently rated as the fastest supercomputer in the world as of this writing [13]. The basic building block of Summit is the IBM Power System AC922 node. Each of the approximately 4,600 compute nodes on Summit contains two IBM POWER9 processors and six NVIDIA Volta V100 accelerators and provides a theoretical double-precision capability of approximately 40 TF. Each POWER9 processor is connected via dual NVLINK bricks, each capable of a 25GB/s transfer rate in each direction. Nodes contain 512 GB of DDR4 memory for use by the POWER9 processors and 96 GB of High Bandwidth Memory (HBM2) for use by the accelerators. Additionally, each node has 1.6TB of non-volatile memory that can be used as a burst buffer. Summit mounts a POSIX-based IBM Spectrum Scale parallel file system called Alpine. It consists of 77 IBM Elastic Storage Server (ESS) GL4 nodes and has a maximum capacity of 250 PB. Each IBM ESS GL4 node is constituted by two dual-socket IBM POWER9 storage servers, and a 4X EDR InfiniBand network for up to 100 Gbit/sec of networking bandwidth. The maximum performance of the final production system is about 2.5 TB/s for sequential I/O and 2.2 TB/s for random I/O.

Gyrokinetic Toroidal Code

As a particle-in-cell code, GTC [1] tracks individual charged particles in a Lagrangian frame in a continuous phase-space, whereas the moments (number density, charge density and current density etc) of particle distribution of different species (thermal ion, thermal electron, fast ion, fast electron, etc.) are simultaneously computed on a stationary Eulerian field mesh. The electromagnetic fields are then solved on the field mesh using proper combinations of Poisson equation, Ampere’s law, Faraday’s law and force-balance equations with finite

difference and finite element methods. This field mesh is also used to interpolate the local electromagnetic fields at the particle positions in phase-space. GTC has been extensively applied to study collisional transport [2], energetic particle transport [3], microturbulence [4], Alfven eigenmodes [5], kink modes, and tearing modes in fusion plasmas.

GTC carried out the first fusion production simulations at tera-scale on the Seaborg computer at NERSC in 2001 [6] and at peta-scale in 2008 as an early science application on the Jaguar computer at OLCF [7] GTC is a key production code in the DOE SciDAC ISEP Center [8] and one of the two fusion codes selected by the Center for Accelerated Application Readiness (CAAR) [9], a DOE ASCR program to prepare prominent codes across all DOE supported research portfolio for the emerging exascale computers. GTC is also one of the production codes in the Summit acceptance benchmark suite.

GTC uses MPI domain decomposition, particle decomposition, and OpenMP shared memory partitioning to scale up to millions of CPU cores to take advantage of the memory hierarchy. Thanks to closed collaborations with computational scientists through ASCR CAAR and SciDAC ISEP projects, GTC has been ported to GPU-based supercomputers including Titan and Summit early access systems at OLCF. The computationally expensive particle and field subroutines are fully ported and optimized on GPU using OpenACC and CUDA. Using realistic fusion simulation parameters [10], GTC shows near-ideal weak scaling performance up to the full capacity of Titan computer with GPU. GTC has recently demonstrated good scalability on more than 20% of Summit and achieved an unprecedented speed of one trillion particle pushes in 2 seconds wall-clock time using 928 nodes on Summit [11].

Preparing GTC for large simulations that generate large volumes of data requires special consideration. GTC generates a combination of diagnostics data, field data, checkpoint-restart files, and particle data. These data components differ vastly in terms of their output frequency, number of variables contained within, and size. Table 1 provides a summary of the various data items along with some of their important characteristics in a typical large simulation.

Table 1: Summary of the different types of data generated by GTC

Data Type	**Frequency of Output**	**Size of the data per output**	**Number of variables**
Diagnostics	Every timestep or every few timesteps	Megabytes	50 - 100
Snapshot	Once every hour	Megabytes up to many Gigabytes	50 - 100
Checkpoint	Once every hour	Terabytes, potentially up to Petabytes	< 20
Restart	Once	Terabytes, potentially up to Petabytes	< 20
Field data	Every timestep	Gigabytes	< 20
Particles	Every timestep	Few hundred terabytes up to petabytes	< 20

In general, diagnostics data are high velocity data that are written frequently, and read back multiple times for analysis, both online as well as offline. Checkpoint data are write-once; that is, they are written for resilience purposes with the intention of simulation restarts. Checkpoint and particle data are high volume and writing them usually consumes significant resources. Furthermore, checkpoint and restart data are high variety data, as their underlying variables are comprised of scalars, vectors, and multi-dimensional arrays.

GTC traditionally uses POSIX I/O, where data are written to binary files. Based on historical knowledge, multiple processes writing data to the same file generally yields poor performance. To circumvent this commonly known N-1 write pattern, all high-volume data in GTC are written such that each process writes its data to an independent file. For many outputs such as field data, data are written into new files per process per output timestep. However, such an approach typically suffers from two issues. One, the metadata overhead for parallel file systems on supercomputers is a massive challenge and often forms the bottleneck for intensive I/O. At extreme scales, this will be one of the most important challenges. Two, a logical file being split into multiple sub-files to improve I/O performance puts additional burden on the developers to ensure that data is read back correctly. Inspecting such data offline can be highly prohibitive without specialized tools.

For the different types of outputs in GTC, optimizing I/O is challenging as there is no 'one-size-fits-all' approach that can be used to optimize I/O. Different strategies need to be adopted for different components. At the exascale, the sheer size of the data can be an issue as many Petabytes of data can be generated over the course of a few hours.

ADIOS

ADIOS is an I/O framework which provides a simple and flexible way for scientists to describe the data in their code. ADIOS provides highly optimized I/O routines that allow users to read and write data in an optimal fashion for the target architecture. In the ADIOS design, variables and steps are first-class concepts. ADIOS provides the API to define a variable which may be a simple scalar or a global array partitioned amongst processes. It provides the ability to write variables to a file and append "steps" to it, so that a single file can contain information about a physical quantity as it evolves over time. ADIOS stores data in a proprietary, log-based, file format named BP (current version 3). For every process that writes data to a file, it creates an independent sub-file and writes metadata in the global file container to reconstruct the original order of the file. The log-based data organization allows ADIOS to write each process's output into a separate chunk of file or aggregate several processes' outputs into a smaller number of files, which can maximize the I/O bandwidth. These operations are kept transparent from the end user. The self-describing nature of the file allows users to inspect the file outside the scope of the running simulation using pre-bundled tools that come with the library.

The ADIOS API is designed as a publish-subscribe library. Applications can write data directly to the underlying storage or publish data so that it can be read by processes that subscribe to it. This allows various applications to couple through in-memory transports, which plays an important role in providing the ability to analyze data in situ. This is done through the use of built-in ADIOS "engines" that provide users with a way to select how data must be published. For example, there are engines to read/write data from/to underlying storage, communicate with other applications through in-memory data exchange, as well as perform a wide area network transfer of data to remote sites. ADIOS also provides the ability to transform data through various compression and reduction methods. Users may set these

options directly in their code or in an XML file. Furthermore, ADIOS provides ways to tune parameters for different engines and operations to optimize I/O performance.

Data Management in GTC using ADIOS

In this section, we discuss our efforts at improving I/O performance in GTC through the use of ADIOS and present results from experiments run on the Summit supercomputer.

At large scale, GTC suffers from issues arising due to sub-optimal I/O patterns and the metadata overhead on the file system. To overcome these limitations, ADIOS is now being used to manage data in GTC. The core design features of ADIOS lead to performance improvements out of the box for various data output by GTC. As timesteps can be appended to existing files, new files per output timestep are no longer required. Additionally, the number of writers, and thus the number of sub-files for different outputs in GTC has been tuned to obtain good performance by alleviating the metadata overhead on the file system. Table 2 lists the number of output files that were created by the original POSIX I/O version of GTC and the ADIOS version of GTC for a run that simulated 10,000 timesteps using 3072 processes on Summit. No particle data was generated was generated for these experiments.

Table 2: Number of output files created by GTC when using POSIX vs. using ADIOS.

Output filename	**Number of files in the POSIX I/O module**	**Corresponding ADIOS output file container**	**Number of files created by the ADIOS module**
equilibrium.out	1	equilibrium.bp	1
data1d.out	1	data1d.bp	1
history.out	1	history.bp	1
snapshot#.out	10,000	snapshot.bp	1
phi#.out	32,000	phi3d.bp	2048
restart1.#	3072	restart1.bp	2048
restart2.#	3072	restart2.bp	2048

It can be seen that the POSIX I/O version of GTC creates almost 50,000 files, whereas the ADIOS version creates just over 6000 files on storage with 4 writers per node, with further potential to reduce the number of sub-files. This leads to significant improvements in the overall I/O performance. This is further demonstrated by the performance improvements in writing snapshot data, as shown in Figure 1. Recall from Table 1 that snapshot data are written to a new file at every output step. Using ADIOS, snapshot data are appended to an existing file which leads to 50x improvement over the POSIX version of the application.

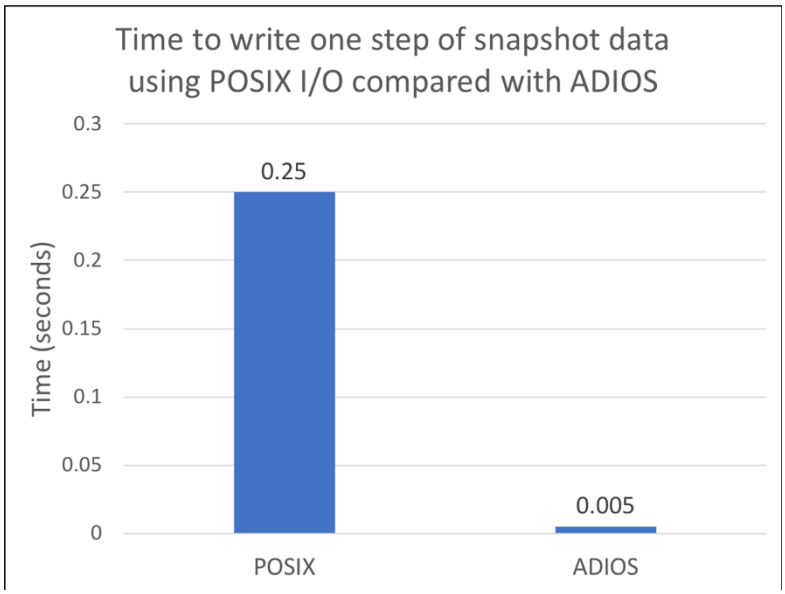

Figure 1: Comparison of the time taken to write one snapshot of GTC on Summit to the GPFS parallel file system for POSIX vs. ADIOS

Figure 2 shows the I/O performance for writing checkpoint data from GTC to the GPFS parallel file system using ADIOS. The simulation was run on 512 nodes with 6 processes per node, for a total of 3072 processes. The aggregate size for each checkpoint was 2.6 Terabytes. For our tests, we compute 5000 simulation timesteps and write 50 checkpoints. We show results for varying number of writers per node in ADIOS. The figure shows that a peak performance of 2.27 TB/s is observed when we use 6 writers per node. Using 6 writers per node displays high variability in I/O performance with an average bandwidth of 1.1 TB/s, whereas using 4 writers per node shows a more consistent average of 1.5 TB/s.

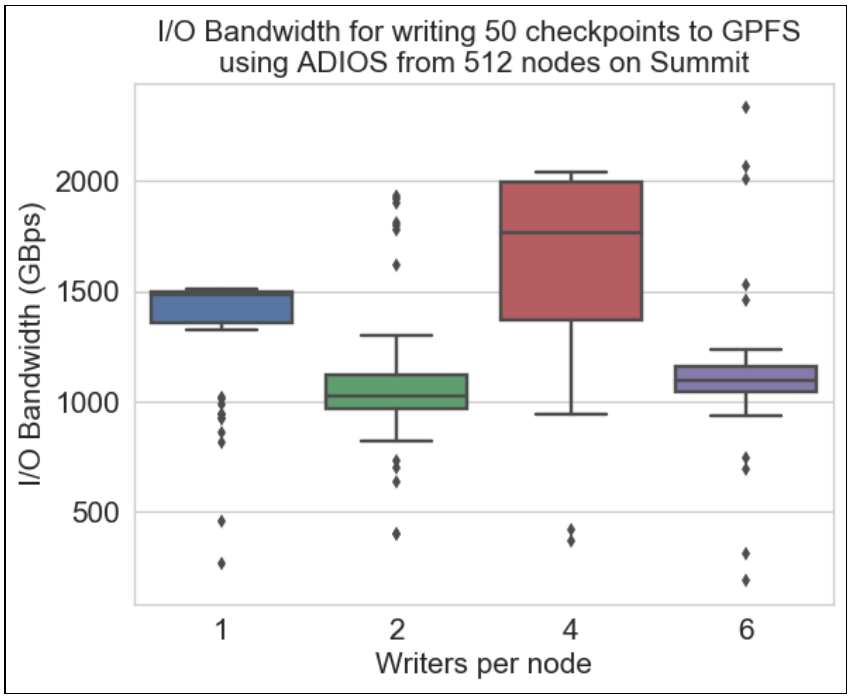

Figure 2: Using ADIOS to write checkpoint data from GTC to the GPFS parallel file system on Summit. Results show the effect of tuning ADIOS options to control the number of sub-files created on the parallel file system for checkpoint data.

I/O variability is common on large-scale computing facilities since the I/O bandwidth is shared by hundreds of running jobs simultaneously. Previous studies [16] [17] have shown significant I/O performance variability on DoE's Titan [18] and Cori [19] supercomputers. This can be mitigated to an extent by the use of the local non-volatile memory on Summit nodes. An in-depth study to model the performance of NVME for use in GTC and effective ways to utilize it is an important future work for this research.

Metadata Optimizations in ADIOS for Large Data

An ADIOS file is a collection of sub-files created by writer processes and metadata information required to recreate the file in its original intended form. With increasingly large output data, the overhead of this metadata is no longer negligible. This will be an important consideration as we move towards the exascale. In this section, a study of the metadata overhead in ADIOS for different types of output data is discussed, along with a design of the next generation file format in ADIOS.

Factors that affect metadata overhead

Experiments have been performed to study the impact of the data size, number of variables, number of processes writing data, and the number of timesteps generated by GTC. To study the impact of data volume, the simulation was run on 64 compute nodes on Summit (6 processes per node) for 100 simulation timesteps. In each step, the size of the variables is varied, while the number of variables is kept constant. As shown in Figure 3, an increase in data size leads to an increase in the overall write time but does not show a significant increase in metadata overhead. This is expected, as ADIOS generates metadata for every write issued to the underlying file by every writer process. Thus, for a constant number of write operations, the metadata overhead remains constant. The increase in overhead seen in the figure is attributed to variability in the I/O bandwidth as the size of the metadata is small. However, from the figure, we can observe that if the size of GTC output data is relatively small, the metadata overhead dominates the total I/O overhead.

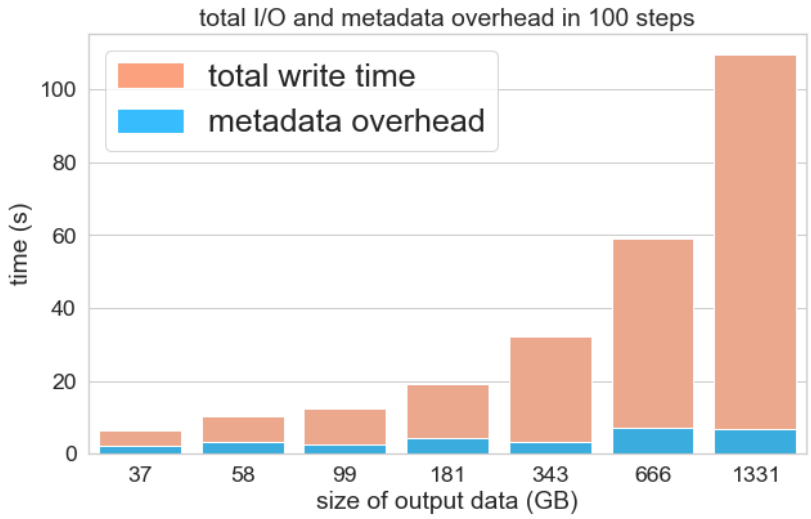

Figure 3: Impact of increasing data volume on metadata overhead

At larger scales, an increase in output data also corresponds to an increase in the number of physical quantities written by the application. Figure 4 shows the impact of an increase in the number of variables generated by GTC. Increasing the number of variables causes a

significant increase in the metadata overhead. The amount of metadata generated increases with an increase in the number of variables, as does the time complexity of constructing and writing those metadata items.

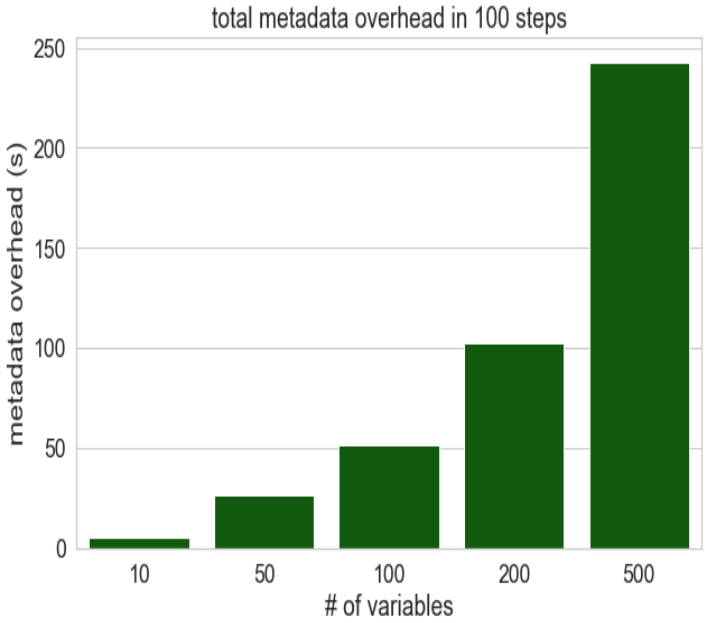

Figure 4: Impact of increasing number of variables on metadata overhead

In order to fully utilize the available I/O bandwidth of parallel file systems, ADIOS allows MPI processes of large-scale simulation runs to write data to separate files, which requires metadata to associate data chunks with processes, and recording the offset of the data chunk in the global file. To study the impact of increasing writers, experiments were run with a constant number of variables and timesteps, but varying number of MPI processes. Figure 5 shows that increasing the number of writers shows almost a linear increase in the metadata overhead.

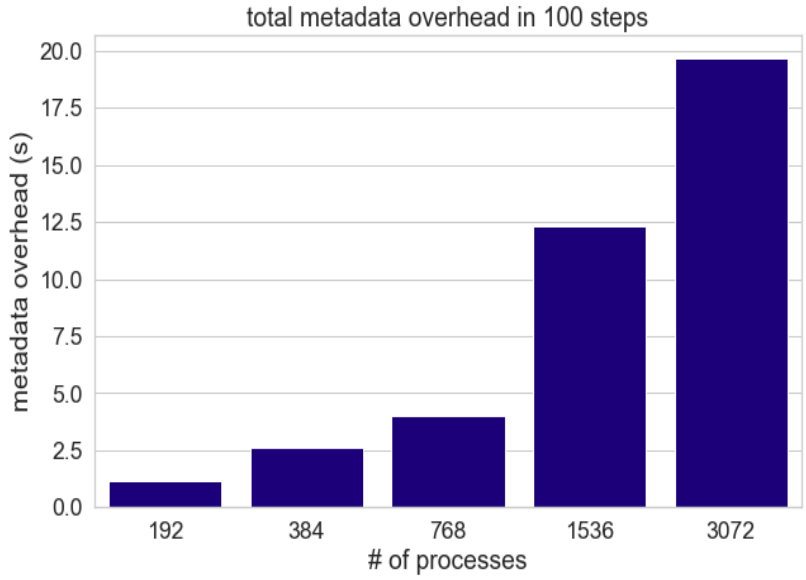

Figure 5: Impact of increasing the number of writer processes on metadata overhead

Recall that applications can append "steps" to an ADIOS file. In addition to metadata associated with variables and writer processes, ADIOS also needs to maintain metadata about the steps in the file. At large scale, GTC can run tens of thousands of time steps. To study the impact of increasing number of timesteps in the simulation, experiments were run with a fixed number of variables, output data sizes, and MPI process count, while increasing the number of timesteps simulated by the application. Figure 6 shows that the average metadata overhead per simulation step increases with an increase in the total number of simulation steps. This is because when ADIOS constructs the metadata at each simulation step, it needs to reorganize and serialize metadata of current and all previous steps in memory, and write the serialized metadata to file.

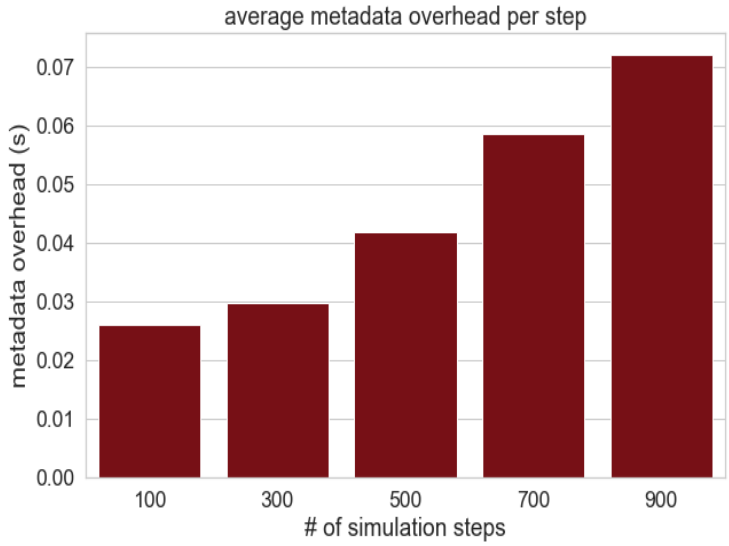

Figure 6: Impact of increasing number of timesteps in GTC on the metadata overhead

When combined with a large number of variables, process counts, and simulation timesteps, the metadata overhead in ADIOS can grow quickly. To alleviate the pressure of increasing number of processes, users can tune ADIOS parameters to reduce the number of writers. However, the overhead due to increasing timesteps is a primary concern as reconstructing the full metadata for every step written to an ADIOS file can lead to an exponential increase in metadata overhead. At extreme scale, this is projected to be a major limiting factor in obtaining good I/O performance. In the next section, we discuss performance improvements obtained through a re-designed metadata format for ADIOS that mitigates this issue.

BP4: The next generation ADIOS file format with improved metadata capabilities

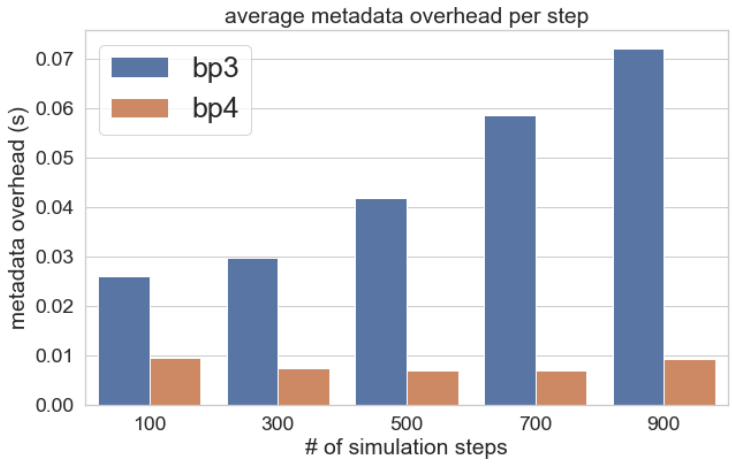

Figure 7: Per-step metadata overhead of current and new metadata construction approach

The BP4 file format is the next generation file format for ADIOS that is targeted towards exascale simulations that generate high volumes of data. Emphasis has been put on redesigning the metadata schema to optimize for increasing timesteps. The central concept is the introduction of an index table that stores the metadata offsets that represent timesteps in the global metadata file. Using an index table removes the need to sort the metadata and parse it serially to retrieve information when timesteps are appended to a file. Figure 7 shows that the metadata overhead per step with BP4 is constant for an increasing number of timesteps. Consequently, the total metadata overhead during the entire simulation run is also significantly reduced.

Conclusions

In this paper, we use Gyrokinetic Toroidal Code (GTC) as a concrete example to demonstrate the data management challenges that arise when we run pre-exascale scientific simulations. We use ADIOS to optimize I/O and data management in GTC. Initial experiments on the Summit supercomputer show a peak performance in excess of 2 TB/s for writing checkpoint data to the GPFS parallel file system. An improvement of 50X is obtained for writing snapshot data in GTC with ADIOS. A novel file format named BP4 for ADIOS is introduced with the objective of reducing metadata overhead at extreme scale. Preliminary results show significant improvement in metadata performance of ADIOS with the new file format. Optimizing I/O on leadership class machines is challenging and further research is required to efficiently utilize the evolving complex storage hierarchy that includes non-volatile memory along with new parallel file systems.

References

[1] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White, Turbulent Transport Reduction by Zonal Flows: Massively Parallel Simulations, *Science* 281, 1835 (1998).

[2] Z. Lin, W. M. Tang, and W. W. Lee, Neoclassical Transport in Enhanced Confinement Toroidal Plasmas, *Phys. Rev. Lett*. 78, 456 (1997).

[3] Wenlu Zhang, Zhihong Lin, and Liu Chen, Transport of Energetic Particles by Microturbulence in Magnetized Plasmas, *Phys. Rev. Lett*. 101, 095001 (2008).

[4] H. S. Xie, Y. Xiao, and Z. Lin, New Paradigm for Turbulent Transport Across a Steep Gradient in Toroidal Plasmas, *Phys. Rev. Lett*. 118, 095001 (2017).

[5] H. S. Zhang, Z. Lin, and I. Holod, Nonlinear Frequency Oscillation of Alfven Eigenmodes in Fusion Plasmas, *Phys. Rev. Lett*. 109, 025001 (2012).

[6] Z. Lin, T. S. Hahm, S. Ethier, and W. M. Tang, Size Scaling of Turbulent Transport in Magnetically Confined Plasmas, *Phys. Rev. Lett*. 88, 195004 (2002).
[7] Yong Xiao and Zhihong Lin, Turbulent Transport of Trapped Electron Modes in Collisionless Plasmas, *Phys. Rev. Lett*. 103, 085004 (2009).
[8] https://www.scidac.gov/projects/2018/fusion-energy-sciences/isep.html
[9] https://www.olcf.ornl.gov/caar/
[10] Zhixuan Wang, Zhihong Lin, Ihor Holod, W. W. Heidbrink, Benjamin Tobias, Michael Van Zeeland, and M. E. Austin, Radial Localization of Toroidicity-Induced Alfven Eigenmodes, *Phys. Rev. Lett.* 111*,* 145003 (2013).
[11] Wenlu Zhang, Wayne Joubert, Peng Wang, Matthew Niemerg, Bei Wang, William Tang, Sam Taimourzadeh, Lei Shi, Jian Bao, and Zhihong Lin, Heterogeneous Programming and Optimization of Gyrokinetic Toroidal Code Using Directives, *Lecture Notes in Computer Science* 11381, 3–21 (2019). (WACCPD 2018 Workshop, Dallas).
[12] https://www.olcf.ornl.gov/for-users/system-user-guides/summit/summit-user-guide/#system-overview
[13] top500.org
[14] C. S. Chang, S. Klasky, J. Cummings, R. Samtaney, A. Shoshani, L. Sugiyama, D. Keyes, S. Ku, G. Park, S. Parker, N. Podhorszki, H. Strauss, H. Abbasi, M. Adams, R. Barreto, G. Bateman, K. Bennett, Y. Chen, E. D. Azevedo, C. Docan, S. Ethier, E. Feibush, L. Greengard, T. Hahm, F. Hinton, C. Jin, A. Khan, A. Kritz, P. Krsti, T. Lao, W. Lee, Z. Lin, J. Lofstead, P. Mouallem, M. Nagappan, A. Pankin, M. Parashar, M. Pindzola, C. Reinhold, D. Schultz, K. Schwan, D. Silver, A. Sim, D. Stotler, M. Vouk, M. Wolf, H. Weitzner, P. Worley, Y. Xiao, E. Yoon and D. Zorin, Toward a First-Principles Integrated Simulation of Tokamak Edge Plasmas*, Journal of Physics: Conference Series,* Volume 125, Number 1 (2008).
[15] http://www.cgl.ucsf.edu/chimera/about.html
[16] Lipeng Wan, Matthew Wolf, Feiyi Wang, Jong Youl Choi, George Ostrouchov and Scott Klasky, Comprehensive Measurement and Analysis of the User-Perceived I/O Performance in a Production Leadership-Class Storage System, *IEEE 37th International Conference on Distributed Computing Systems* (2017).
[17] Lipeng Wan, Matthew Wolf, Feiyi Wang, Jong Youl Choi, George Ostrouchov and Scott Klasky, Analysis and Modeling of the End-to-End I/O Performance on OLCF's Titan Supercomputer, *IEEE 37th International Conference on Distributed Computing Systems* (2017).
[18] https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/
[19] https://www.nersc.gov/users/computational-systems/cori/
[20] Lofstead, J. F., Klasky, S., Schwan, K., Podhorszki, N., & Jin, C. (2008, June). Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments* (pp. 15-24). ACM.
[21] Fu, H., Liao, J., Yang, J. et al. Sci. China Inf. Sci. (2016) 59: 072001. https://doi.org/10.1007/s11432-016-5588-7
[22] https://computation.llnl.gov/computers/sierra