**PAPER** • **OPEN ACCESS**

# Organizing Large Data Sets for Efficient Analyses on HPC Systems

To cite this article: Junmin Gu *et al* 2022 *J. Phys.: Conf. Ser.* **2224** 012042

View the article online for updates and enhancements.

ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
241st ECS Meeting
Register now!
Vancouver, BC, Canada. May 29 – June 2, 2022
ECS Plenary Lecture featuring
Prof. Jeff Dahn,
Dalhousie University

# Organizing Large Data Sets for Efficient Analyses on HPC Systems

**Junmin Gu[1], Philip Davis[2], Greg Eisenhauer[3], William Godoy[4], Axel Huebl[1], Scott Klasky[4], Manish Parashar[2], Norbert Podhorszki[4], Franz Poeschel[5], Jean-Luc Vay[1], Lipeng Wan[4], Ruonan Wang[4], and Kesheng Wu[1,*]**

[1] Lawrence Berkeley National Laboratory, USA
[2] Scientific Computing and Imaging (SCI) Institute, University of Utah, USA
[3] Georgia Institute of Technology, USA
[4] Oak Ridge National Laboratory, USA
[5] Center for Advanced Systems Understanding and Helmholtz-Zentrum Dresden-Rossendorf, Germany

*Email: KWu@lbl.gov

**Abstract.** Upcoming exascale applications could introduce significant data management challenges due to their large sizes, dynamic work distribution, and involvement of accelerators such as graphical processing units, GPUs. In this work, we explore the performance of reading and writing operations involving one such scientific application on two different supercomputers. Our tests showed that the Adaptable Input and Output System, ADIOS, was able to achieve speeds over 1TB/s, a significant fraction of the peak I/O performance on Summit. We also demonstrated the querying functionality in ADIOS could effectively support common selective data analysis operations, such as conditional histograms. In tests, this query mechanism was able to reduce the execution time by a factor of five. More importantly, ADIOS data management framework allows us to achieve these performance improvements with only a minimal amount of coding effort.

## 1. Introduction

Many recent scientific discoveries are from large-scale data analyses [1, 2]. This work examines the input/output, I/O, strategies to read/write the large amounts of data supporting these scientific discoveries. Because the scientific data is typically stored as multi-dimensional arrays in large files, we specifically focus on strategies that enable efficient read operations on these files. This work focuses on one of the I/O libraries, the Adaptable Input Output System, ADIOS (https://github.com/ornladios/ADIOS), because it requires relatively little tuning to achieve good write performance. This choice allows us to pay more attention to the read performance during data analyses.

ADIOS is relatively new among parallel I/O libraries. Its design incorporates key lessons from the earlier systems to make the software easier to use [3, 4]. For example, most existing I/O libraries require users to describe data carefully in their source code, where any change to the I/O operations would require the program to be modified and recompiled, while ADIOS considerably simplified this description by separating the backend "engines" from the unified application programming interface (API) exposed to the library consumer. In addition, ADIOS could be adjusted through a runtime configuration file to modify tuning parameters, transport mechanisms, and so on, which further reduces the coupling between user code and ADIOS.

Based on various published studies, we know that scientific data is largely represented by multi-dimensional arrays [5, 6, 7]. Many of the these arrays are from simulations employing regular meshes to discretize their problem domains, however, to scale up simulations to new limits (*e.g.* exascale systems), they are switching to dynamic meshing mechanisms, such as adaptive mesh refinement (AMR) techniques [8]. Thus, it is timely for us to explore the performance tuning with one of these AMR based simulations at scale. In this study, we have chosen the state-of-the-art plasma physics simulation code WarpX [9]. The plasma particles in this simulation could move from processor to processor, adding to the dynamic meshes introduced by AMR, this code could have complex I/O patterns with significant load imbalances [10]. To abstract out the computational aspects from the physical particularities of this application and to study the impact of this load imbalances, we also created a synthetic benchmark based on the schema of the WarpX data.

In a typical data analysis scenario, only a relatively small fraction of a data set is needed. In this work, we use a set of scenarios from a WarpX simulation of laser-driven accelerator to demonstrate the ADIOS querying capability for supporting selective data accesses. The previous version of ADIOS has an efficient metadata strategy that could be considered as a block index for its querying mechanism [7]. We have been working on reintroducing this block index and the query support in the new version of ADIOS [3]. This work reports our experience of working with this updated querying mechanism on a new generation of simulation data.

The main contribution of this work includes:

- A systematic study of I/O performance of an AMR-based plasma physics simulation code WarpX. Because WarpX employs dynamic data structures and makes extensive uses of accelerators, it examplifies a new generation of scientific simulation codes.
- A synthetic benchmark to help study the imbalances in the I/O operations introduced by the dynamic data structures.
- A study of the querying mechanism in the latest version of ADIOS library.

## 2. Background

Before describing our experience of tuning the options for organizing files for efficient I/O operations, we describe some background information, including the parallel I/O ADIOS library, the particle data standard openPMD, the HPC AMR simulation WarpX code, and the benchmark suite used to generate synthetic I/O test operations.

ADIOS is a framework for large-scale scientific data management tasks including data generation transfer, and storage across multiple transport media using a unified API [3, 4, 11]. It is well-known for obtaining near-optimal write-performance for simulations [12]. A notable feature is its *in situ* processing capability [13]. These features make ADIOS an attractive option for many science applications under the Exascale Computing Project (ECP). In this paper, we will only evaluate a few of the most efficient I/O operations, for example, having each compute node or having each message passing interface, MPI, process rank work independently on their I/O operations. For a thorough description of ADIOS, we refer readers to some published studies [3].

Many HPC simulations are capable of producing a large amount of output. Among them, particle-in-cell (PIC) codes are well-known because they frequently track trillions of particles in complex electromagnetic fields [12, 14]. Their checkpoint files could easily reach hundreds of terabytes. In addition, it is hard to distribute both particles and mesh cells evenly on compute nodes, which leads to uneven I/O operations with suboptimal I/O performance.

To represent the complex electromagnetic field, a number of PIC simulations are starting to refine their field representation adaptively [8], which increases the I/O complexity. To cope with this increased complexity, developers of PIC simulations are consolidating their I/O operations with the common API known as openPMD-api [15]. We use openPMD for I/O operations because it allows us to control the format of data files as well as the data access patterns.

Our tests involve a set of synthetic benchmarks as well as a state-of-the-art PIC code named WarpX [9]. The synthetic benchmarks are constructed following typical PIC codes, but contain random values that are evenly distributed among the writers. This uniform data distribution reduces the I/O variability and simplifies performance studies. The data distribution from WarpX is uneven. We choose WarpX as an example from a new generation of simulation codes because it is among the first to embrace AMR for mesh management and make extensive use of GPUs in computation. Thus, it would allow us to observe potential I/O performance characteristics for the next generation of GPU-based simulations.

## 3. File Organizations for Efficient I/O

Our evaluation on how to organize files for efficient analyses is divided into two parts: one with synthetic benchmarks and the other with WarpX. These evaluations are conducted on supercomputers named Summit and Cori. Summit, located at Oak Ridge National Laboratory, is developed by IBM and has 200 petaFLOPS peak performance [16]. The file system we use on Summit is Alpine, a 250 PB IBM Spectrum Scale GPFS file system (https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/summit-faqs/). It has peak performance of roughly 2.5 TB/s of I/O. The Cori supercomputer is a Cray XC40 system with a theoretical peak performance of 30 petaFLOPS (https://www.nersc.gov/systems/cori/). The file system we use on Cori is Lustre (https://docs.nersc.gov/performance/io/lustre/). In all reported tests we use 32 object storage targets (OST) and 32MB stripe size. The report I/O performance measurements are based on the timers around the I/O operations, averaged over all participating computer cores when more than one is involved.

We start our performance exploration with a set of synthetic benchmarks. It has a writer and a reader, to generate and retrieve 1D, 2D, and 3D particle and mesh data, following the openPMD schema [17]. The data is organized by timesteps where each timestep has 10 field and particle variables, similar the later WarpX tests. In the following tests, each run of the writer produces 10 timesteps, on 3D meshes. Each run of the writer starts with 64 nodes (16 MPI ranks per node) to generate 6.4TB of data and about 10MB of metadata. We increased the output size with the number of processes used (i.e. weak scaling) on Summit, while kept the total size constant on Cori (i.e. strong scaling).

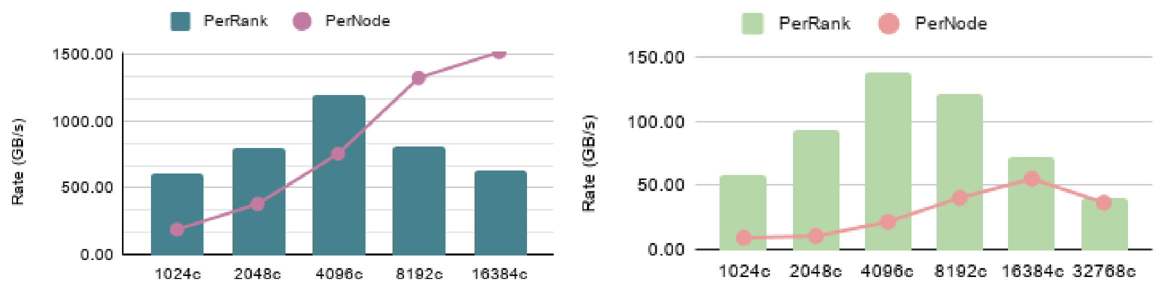

**Figure 1.** Weak-scaling performance of two ADIOS writing options, one subfile per MPI rank and one subfile per compute node. These options reduce the dependencies among the compute nodes and are typically the most performant options.

In the write tests, we report two options are known to perform well: (1) writing one subfile per MPI rank and (2) writing one subfile per compute note. Since these subfiles can be written independently ADIOS achieves high bandwidth on storage systems. However, each subfile is actually a file on the file system and the file creation is relatively expensive on a file system, if a large number of subfiles are created, then the overall performance might decrease. One concrete objective for our study is to determine how many subfiles might be too many.

As shown in Figure 1 for both Summit and Cori systems, using a relatively modest number of MPI ranks, say, 1024, the one subfile per rank option produces faster I/O rates. This trend stops after 4K cores on Summit, and 8K cores on Cori. With more MPI ranks, the overhead of creating files becomes larger than time needed to aggregate the write operations on each compute node, which leads the one subfile per node to be the best performing option.

The benchmark reader implements several common read patterns [6]. In this paper, we only report two timing measurements: metadata loading time and retrieving a plane from a 3D array. Following the terminology used in WarpX, the plane we retrieve is the YZ plane, which is contiguous had the 3D array been organized in a single large memory space.

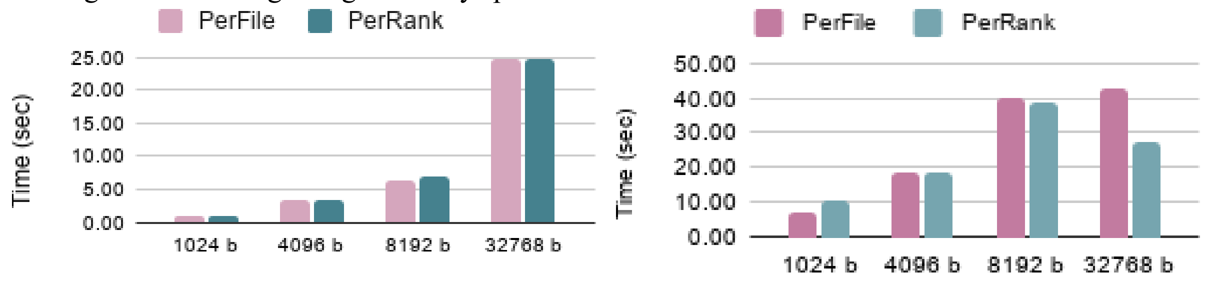

**Figure 2.** Time (in seconds) needed to complete some read operations on a single core on Summit versus the number of blocks in a single ADIOS variable. We see that the time needed generally increases as the number of blocks increases.

Data in applications often come in batches from each processor, which is called a "block" in ADIOS. The number of blocks impacts ADIOS I/O throughput, as shown in Figure 2. In these tests, we kept the same number of subfiles and same global array dimensions, but varied block sizes on disk for each variable. In other words, a processor can store all its content all at once (1 block), or in many batches (many smaller blocks). The metadata size increases linearly as the number of blocks grows, which in turn increases metadata loading time on both Cori and Summit (though only Summit measurements are reported in Figure 2). From the read time shown in Figure 2, we see that the time to complete both read operations generally increases with the number of blocks. Even though the YZ plane is logically contiguous, the content is actually distributed into many blocks.

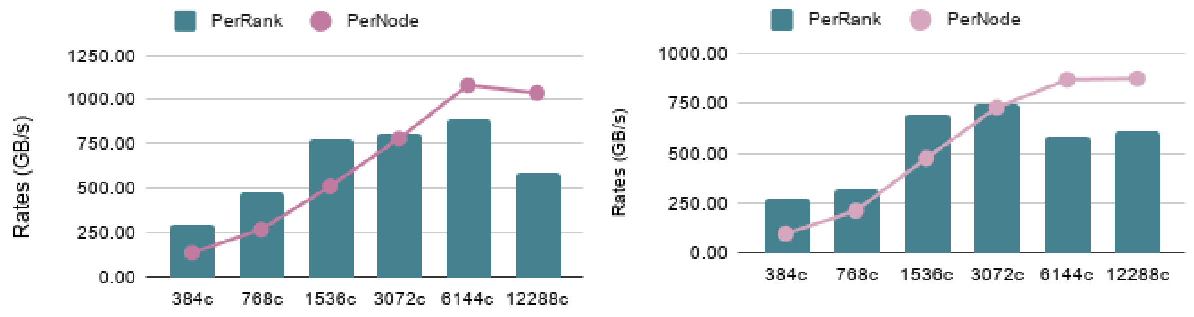

**Figure 3.** WarpX write performance on Summit. The two load balancing strategies, Knapsack and Z order, show similar weak-scaling performance with the two writing options, subfile per node and subfile per rank.

We next report our performance measurements with WarpX, both the write and read operations. Note that in WarpX, the data sizes vary from time step to time step, and from processor to processor. Additionally, there are more blocks per variable in WarpX than in the synthetic benchmark.

On Summit, WarpX is set up to use all 6 GPUs per computing node. We ran from 64 nodes to 2048 nodes, with output file sizes ranging from 3.9TB to 125TB (i.e., weak scaling). We focus our tests on Summit to understand how GPUs might affect I/O performance.

Although resource configurations and workloads differ, what we observed in Figure 3 is close to what we saw on the synthetic benchmarks. WarpX groups its computational blocks (defined by AMReX) using two different strategies for load balancing: Knapsack and Z order. Using either strategy, one subfile per rank option out performs the one subfile per node option until 6144 ranks. As the number of

MPI ranks increases further, the second option performs better. With one subfile per node option, the best rate we achieved is 1TB/sec on summit (6144 ranks with Knapsack load balancing).

The WarpX output files contain more blocks per variable. Therefore, the size of metadata is much bigger and requiring more time to load, see Figure 4. However, the read time consumed for different reading patterns is similar to that of the synthetic benchmark. Furthermore, the two load balancing methods did not show any significant impact on reading performance.

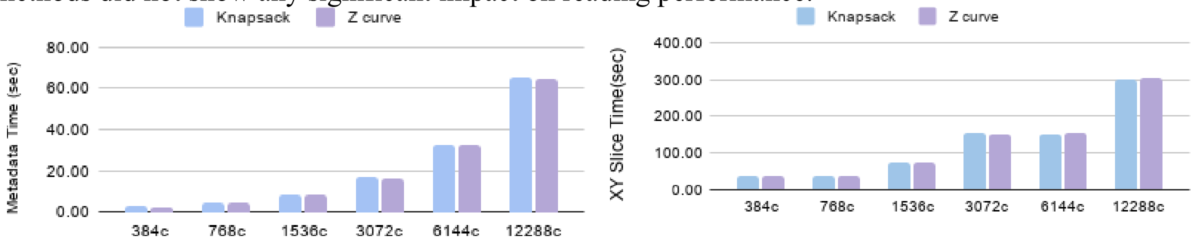

**Figure 4.** Time (in seconds) to complete some read operations on WarpX data on Summit (plotted aganst the number of compute cores involved).

## 4. Supporting Efficient Filtering

Many analysis operations only need to access data from a small part of the simulation domain or to satisfy some other user-defined conditions. This type of filtering (or querying) is particularly useful when the data collection is very large, as in the case with some large scale WarpX runs. In some cases, there could be $10^{13}$ particles along with $10^9$ mesh cells for representing fields. Additionally, some parts of the mesh could be refined multiple times creating complex data structures that are especially challenging to handle efficiently. Without an efficient filtering strategy, the data analysis functions will have to read the whole data set, which would be very time-consuming [12].

In this work, we have adopted a number of strategies to address this type of selective data access. The first strategy is applying subfiling, as mentioned earlier. Within each subfile, the user data is organized into blocks that serve as the basic unit of I/O operations. Within each block, we maintain a small amount of metadata that includes a version of the block index in the data [7]. This block index effectively records the statistics for values of each variable in a block. By examining this block index, we could decide whether or not a particular block needs to be retrieved before actually committing the effort to perform the necessary I/O operations.

Next, we use a specific WarpX as an example to demonstrate the effectiveness of our approach for supporting selective accesses. In an application scenario, WarpX is used to model the acceleration of particles in a laser-driven accelerator, hence to analyze and correlate accelerated particle beams with a specific accelerator geometry, sub-selecting the particles is a common task. Depending on the exact geometry and plasma response, scientists may select particles emitted below a given angle, in a certain acceptance range, or a part of a certain sub-ensemble. A scientist might vary the selection conditions to explore the characteristic signature of the laser-plasma interaction. In these cases, being able to select the particles satisfying the user-specified conditions quickly is critical to the scientific understanding of the accelerator design.

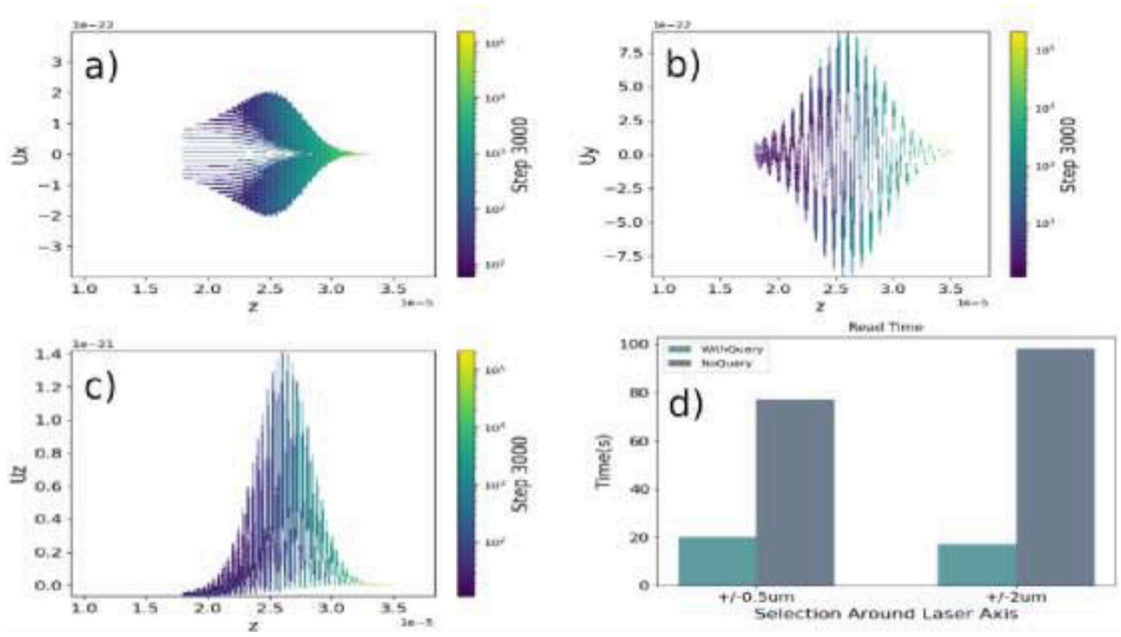

**Figure 5.** Data analysis using region of interest filtering with ADIOS queries. a)-c) Phase space projections of plasma particles oscillating in a laser pulse, filtered close to the laser axis. d) Read time comparison between conventional reads and pre-filtered reads with queries.

Figure 5 shows a common scenario from this accelerator design involving filtering, where the subfigures a)-c) show the histograms of particles in three different phase spaces. This is effectively a mapreduce operation, which is ubiquitous in data science. The particles are selected using the block index embedded in the metadata of the blocks in ADIOS files. In this specific case, the selection is based on the position of particles in 3D (real) space. In sub-figure 5 d), we show a comparison of execution time of generating these histograms with and without the query support. In these cases, we are able to reduce the histogram computation time from 80-100 seconds to about 20 seconds, clearly demonstrating the usefulness of the block index mechanism. (Note that the two NoQuery measurements should require the same amount of time, the observed variations are due to file-system variance.)

## 5. Conclusion and Future Work

We have experimented with a number of promising strategies to optimize I/O rates for large scientific applications on HPC, using ADIOS. These options are primarily exploring the number of subfiles to use and the number of blocks for variables, where the former decides the rate to store data on disk and the later affects the rate to retrieve information back from disk.

In our tests, both the synthetic benchmark and a scientific simulation are able to write at scale on Alp ine at 1TB/sec. We are interested in further studying this simulation code and see how it performs with more levels of mesh refinement.

In this paper, we also report an exploration of the block indexing feature built into the ADIOS file structure. This index structure is able to effectively support selective accesses to the data file, for example, by allowing querying software to determine whether any records in a block satisfy the user-specified conditions. If none of the records in a block could satisfy the conditions, there is no need to actually read the data block from the storage media. Tests showed that we are able to significantly reduce the time needed to compute common tasks such as 2D histogramming (mapreduce). We are interested in studying the options to support more complex query conditions.

To ensure that data analysis operations could read files efficiently, it is important to limit the number of blocks to read. One effective strategy for this is to utilize *in situ* processing capabilities to reorganize the data for more efficient read operations. While we have studied strategies to reduce the number of blocks elsewhere [18], we have also explored *in situ* processing capabilities [19]. We plan to study this *in situ* option further because it is likely to become more important in future exascale systems.

**Acknowledgements**
This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725 and of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. This work was partially funded by the Center of Advanced Systems Understanding (CASUS), which is financed by Germany's Federal Ministry of Education and Research (BMBF) and by the Saxon Ministry for Science, Culture and Tourism (SMWK) with tax funds on the basis of the budget approved by the Saxon State Parliament.

**References**
[1] Clark J, Cadonati L, Healy J, Heng I, Logue J, Mangini N, London L, Pekowsky L and Shoemaker D, 2015. *Gravitational Wave Astrophysics* (Springer) pp 281–287
[2] Hey T, Tansley S and Tolle K 2009 *The Fourth Paradigm: Data-Intensive Scientific Discovery* (Microsoft)
[3] W. F. Godoy, et al. 2020. *SoftwareX* **12** 100561. DOI: 10.1016/j.softx.2020.100561
[4] Liu Q, Logan J, Tian Y, Abbasi H, Podhorszki N, Choi J Y, Klasky S, Tchoua R, Lofstead J, Oldfield R *et al.* 2013. *Concurrency and Computation: Practice and Experience*
[5] Chang C S, et al. 2008. *Scidac 2008: Scientific Discovery through Advanced Computing* **125** 12042–12042
[6] Lofstead J, Polte M, Gibson G, Klasky S, Schwan K, Oldfield R, Wolf M and Liu Q 2011 *Proceedings of* HPDC '11 (ACM) pp 49–60 ISBN 978-1-4503-0552-5. DOI: 10.1145/1996130.1996139
[7] Gu J, Klasky S, Podhorszki N, Qiang J and Wu K. 2018. *Asian Conference on Supercomputing Frontiers* (Springer) pp 51–69
[8] Zhang W, Almgren A, Beckner V, Bell J, Blaschke J, Chan C, Day M, Friesen B, Gott K, Graves D *et al.* 2019. *Journal of Open Source Software* **4** 1370–1370
[9] Vay J L, Huebl A, Almgren A, Amorim L D, Bell J, Fedeli L, Ge L, Gott K, Grote D P, Hogan M, Jambunathan R, Lehe R, Myers A, Ng C, Rowan M, Shapoval O, Thévenet M, Vincenti H, Yang E, Zaim N, Zhang W, Zhao Y and Zoni E 2021 *Physics of Plasmas* **28** 023105
[10] Tang H, Koziol Q, Byna S, Mainzer J and Li T 2019 *2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW)* (IEEE) pp 11–19
[11] Tian Y, Klasky S, Abbasi H, Lofstead J, Grout R, Podhorszki N, Liu Q, Wang Y and Yu W 2011 *2011 IEEE International Conference on Cluster Computing* (IEEE) pp 93–102
[12] Huebl A, Widera R, Schmitt F, Matthes A, Podhorszki N, Choi J Y, Klasky S and Bussmann M 2017 *High Performance Computing* ed Kunkel J M, Yokota R, Taufer M and Shalf J (Springer) pp 15–29 ISBN 978-3-319-67630-2
[13] Bauer A C, Abbasi H, Ahrens J, Childs H, Geveci B, Klasky S, Moreland K, O'Leary P, Vishwanath V, Whitlock B and Bethel E W 2016 *Computer Graphics Forum* ISSN 1467-8659
[14] Nakamura T and Daughton W 2016 3-D VPIC simulation of a vortex-induced reconnection event observed by MMS *Tech. Rep.* 1395321 OSTI.gov
[15] Huebl A, Poeschel F, Koller F, Gu J 2018 *openPMD-api: C++ & Python API for Scientific I/O with openPMD*. 2021. DOI: 10.14278/rodare.27
[16] Oral S, Vazhkudai S S, Wang F, Zimmer C, Brumgard C, Hanley J, Markomanolis G, Miller R,

Leverman D, Atchley S and Larrea V V 2019 End-to-end i/o portfolio for the summit supercomputing ecosystem URL https://doi.org/10.1145/3295500.3356157

[17] Huebl A, Lehe R, Vay J L, Grote D P, Sbalzarini I, Kuschel S, Sagan D, Pérez F, Koller F and Bussmann M. 2015. openPMD: A meta data standard for particle and mesh based data. URL https: //doi.org/10.5281/zenodo.591699

[18] L. Wan *et al.*, 2022. Improving I/O Performance for Exascale Applications Through Online Data Layout Reorganization, in *IEEE Transactions on Parallel and Distributed Systems*, **33**(4), pp. 878-890, doi: 10.1109/TPDS.2021.3100784.

[19] F. Poeschel, et al. 2021. Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2. *arXiv preprint* arXiv:2107.06108