DAI: How pre-computation speeds up data analysis

Kira Duwe[1][0000−0003−3172−1225] and Michael Kuhn[2][0000−0001−8167−8574]

[1] EPFL, Lausanne, Switzerland
[2] Otto von Guericke University Magdeburg, Germany

Abstract. As data sizes continue to expand, the challenge of conducting meaningful analysis grows. Utilizing I/O (Input/Output) libraries, such as HDF5 (Hierarchical Data Format) and ADIOS2 (Adaptable IO System), facilitates the filtering of raw data, with prior research highlighting the advantages of dissecting these formats for enhanced metadata management. Our study introduces a novel data management technique aimed at boosting query performance for HPC analysis applications through the automatic precomputation of commonly used data characteristics, as identified by our user survey. The Data Analysis Interface (DAI), developed for the JULEA storage framework, not only enables querying this enriched metadata but also shows how domain-specific features can be integrated, demonstrating a potential improvement in query times by up to 22,000 times.

Keywords: Scientific data management · Self-describing data formats · HPC applications · Pre-computation.

1 Introduction

Growing computing power and sophisticated data analysis techniques have advanced high-resolution climate research, despite storage and network limitations emerging as significant bottlenecks. While strategies like data compression and I/O optimizations can offer relief, the uneven pace of hardware advancements demands a rethinking of storage systems [5, 8, 23]. Emerging technologies like NVRAM and NVMe SSDs promise enhanced performance but face constraints in cost and capacity, underscoring the need for efficient data management in massive archives like the 300 PB at the German Climate Computing Centre. Addressing these challenges is vital for sustaining long-term access to critical data.

Besides the hierarchy of storage hardware, the I/O software stack also interferes with efficient data sifting and the optimal utilisation of the cluster. In climate research, the applications typically directly employ I/O libraries such as NetCDF (Network Common Data Form) [17], HDF5 (Hierarchical Data Format) [2] and ADIOS2 (Adaptable IO System) [13]. They offer rich metadata and optional hints for the I/O libraries to allow performance tuning. Due to the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

complex interplay of different components and optimisations in the I/O stack, performance issues are common.

No application changes. Developing efficient management techniques for HPC applications faces significant challenges, primarily due to the legacy code base. Scientific simulations are often developed by domain scientists, prioritizing domain knowledge over software engineering principles. Combined with its large size, this poses difficulties in adapting and implementing novel management techniques. Furthermore, asking for application changes is often met with resistance of varying degrees. This is not done out of bad faith but to preserve the considerable time and financial investments made into the code development often over decades. Solutions requiring application changes are unlikely to be tested, much less employed, on a large scale. Unfortunately, this heavily limits the design space for new approaches. In order to offer improvements that can be employed realistically, we focused on making our approach transparent for the application layer.

Contributions. Following the quote from Jim Gray, "Metadata enables data access", we examined the I/O libraries and their corresponding data formats for improvements in the data and metadata management [7]. We think there is great unused potential in these formats, especially ADIOS2. In this paper, we offer a new data management technique to tackle the challenge of finding new insights into PB of data. We build on previous work and dissect the file format inside the I/O libraries, making the changes transparent to the application layer. We extend the previous work to the automatic pre-computation of additional data characteristics that are common in post-processing across different scientific domains, as shown in our user survey. Because we focus on reductions, like sum and mean, the computation and storage overhead is minimal. To query the new metadata, we design the data analysis interface (DAI). Besides retrieval functionality, we show how offering domain-specific functionality like the climate indices can be incorporated, as well as tagging interesting data parts. We make the following contributions:

- Identifying opportunities to improve data management in HPC systems without requiring application changes
- Designing the DAI to allow for the pre-computation of data characteristics, enabling metadata tagging and access at varying levels of granularity, implemented in two ADIOS2 engines (KV-engine and DB-engine)[3].
- Implemented the DAI interface within the JULEA framework, significantly enhancing analysis application query times by up to 22,000 times[4].

2 Observations

The design of the proposed system was influenced by several key observations we made regarding HPC systems and their applications.

[3] https://github.com/parcio/adios2
[4] https://github.com/kiraduwe/julea

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

Data Access in HPC systems. In HPC, checkpointing is a common practice where applications save their current state to resume computations later, especially since scientific simulations may run longer than the time allocated on a cluster. Besides, many HPC workloads follow a write-once-read-many (WORM) pattern, contrasting with database systems that frequently update data. A study of CERN workloads, analyzing over 2.49 billion events, highlighted this pattern, showing minimal updates among the events and a substantial difference between total written data (over 150 PB) and read data (more than 300 PB), emphasizing the predominance of reading over writing [16].

Observation 1: HPC data is predominantly written once, barely updated and read often [19]. Therefore, we focus on improving the read performance, specifically emphasising the post-processing and analysis phase.

HPC Applications. These applications fall into two main categories: large-scale simulations, like climate models, and analysis applications that process data from these simulations. Analysis applications, often crafted by small teams or individual developers, tend to be less complex, have fewer lines of code, and lack extensive optimization or parallelization. This makes them notably more adaptable to code modifications and the adoption of new APIs. *Observation 2: The stark reluctance to accept application changes does not apply to analysis applications, making them an ideal leverage point for trying new approaches and using novel interfaces.*

Self-describing data formats. I/O libraries like HDF5, NetCDF, and ADIOS2, which facilitate easy data management and exchange through self-describing data formats (SDDFs), enable researchers to annotate data with additional attributes, e.g. about experiment runs. SDDFs can be split into *file data* and *file metadata*, as shown in Figure 1. The latter encompasses annotations, such as user-set attributes, as well as structural information about the data, such as variable dimensions and hierarchical ordering within groups. Separating file metadata from data is essential due to their differing access patterns; data servers optimized for large, contiguous I/O are less efficient with the small, random accesses required for metadata, which is currently stored within files, limiting performance.

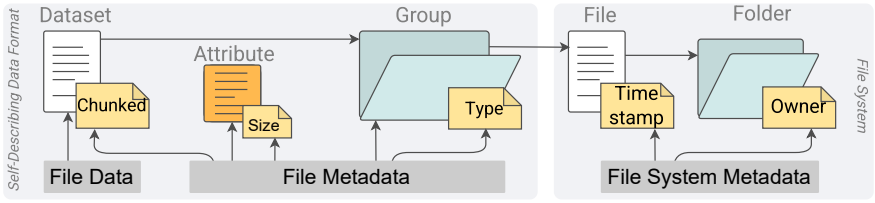


Fig. 1: Different types of metadata are indicated: File system metadata and file metadata. The former is further split into attributes set by the user directly (orange) and structural information encoded in the format (yellow).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

Previous studies have investigated various methods, including using relational databases, duplicating metadata in databases alongside files (e.g., EMPRESS 1 and EMPRESS 2), and eliminating the file concept to exclusively store data in databases and object stores, aiming to improve the accessibility of the file metadata [4, 10, 11]. Further details on these approaches will be discussed in Sections 3 and 4.

Observation 3: Although the separate treatment of file metadata is a positive step, we believe there is untapped potential, particularly in ADIOS2. Expanding upon its capability and pre-computing additional statistics can bring significant benefits.

Typical post-processing operations. To determine appropriate statistics for pre-computation, we examined climate research simulations and their analysis applications. In particular, we looked at the widely used climate data operators (CDOs) [18]. Additionally, we conducted a survey among scientists from different domains to identify common mathematical operations used in data analysis.

Observation 4: Through our analysis, we observed that various domains share typical mathematical operations like reductions, e.g. sum and mean. Therefore, we pre-compute these statistics to improve data retrieval and analysis efficiency.

3 Format Dissection with JULEA

Designing and implementing a distributed storage solution for HPC clusters is complex, often involving custom client/server architectures like EMPRESS [11]. To streamline our efforts and capitalize on existing functionality, we chose to build upon the JULEA framework. JULEA not only reduced implementation work but also provided benchmarking tools, extensive testing, and continuous integration. Operating entirely in user space, JULEA simplifies development and debugging, offering various components like clients, servers, and backends to construct custom storage systems, as depicted in Figure 2. JULEA provides versatile interfaces for interacting with clients directly or integrating into I/O libraries like HDF5 and ADIOS2 through VOL plugins or engines. Supporting object, key-value, and relational database servers, JULEA enables easy adaptation to various access patterns without altering the application code, allowing users to focus on functionality while seamlessly transitioning between backend technologies like LMDB and LevelDB with minimal configuration changes.

The dissection of self-describing data formats using the JULEA storage framework is illustrated in Figure 2. The parallel application communicates across the different compute nodes through MPI. It uses an I/O library such as ADIOS2 or HDF5 for writing and reading its data. The data format is then dissected inside the library and its parts are directed forward to the storage solution by either the engine or the VOL plugin, depending on the used library.

4 Related Work

The Extensible Metadata PRovider for Extreme-scale Scientific Simulations (EMPRESS) stands out as a notable approach to enhanced metadata management,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

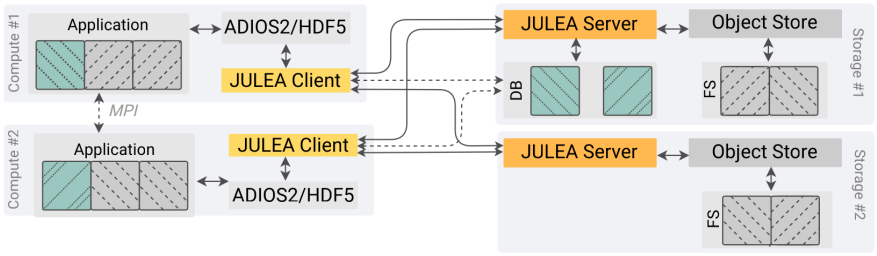


Fig. 2: JULEA setup with two compute nodes and two storage nodes. The key-value store manages the small file metadata pieces (green with fine-grained pattern), while the object store backend handles the large data chunks (grey with coarse-grained pattern). It also distributes the chunks over several storage nodes.

offering a metadata service tailored for self-describing data formats like HDF5 and ADIOS2 [11]. However, EMPRESS and its successor, EMPRESS 2, necessitate explicit specification of all desired metadata information, requiring application changes. In contrast, our solution builds upon the data model introduced by I/O libraries, concealing changes and reducing user involvement significantly.

Moreover, the drawbacks of maintaining a separate database for metadata, as discussed by Zhang et al., underscore the challenges inherent in such approaches [3, 25]. Projects like *BIMM* [9] and *SPOT* [22] also grapple with similar issues, while solutions like *JAMO* attempt to mitigate them using document databases like MongoDB. However, these approaches often encounter duplication of metadata and struggle to model the hierarchical nature of self-describing data formats effectively. By leveraging JULEA, our solution avoids these pitfalls, enabling seamless metadata management without the need for complex transformations or duplication of metadata. Additionally, innovations like *MIQS* [25] aim to address these challenges by introducing schema-free indexing solutions, but they still face limitations such as redundant storage of attribute-file path relations.

The Fast Forward Storage and IO (FFSIO) project aims to revolutionize exascale storage systems to meet the demands of HPC applications and large-scale data workloads. Despite offering a rich I/O interface, DAOS faces limitations due to extensive requirements for NVRAM and NVMe devices, making it unsuitable for cost-constrained environments [12].

The European Centre for Medium-Range Weather Forecasts (ECMWF) contributes to data archival through the ECMWF's File Storage system (ECFS) and Meteorological Archival and Retrieval System (MARS) systems, catering to the needs of weather modeling researcherss [6]. While ECFS manages files primarily with write calls, MARS operates as an object store, providing efficient storage management through a database-like interface.

In the realm of metadata management, projects like DAMASC [1], SoMeta [21] and HopsFS [15] present diverse approaches. DAMASC focuses on redefining the FS interface using database mechanisms, while SoMeta provides metadata

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

infrastructure using a distributed hash table. HopsFS overcomes scalability limitations with distributed metadata services. Furthermore, object-based storage systems have shown efficiency and scalability advantages over traditional I/O stacks, with object-centric storage systems improving HDF5 performance when datasets are stored in object stores [14]. Additionally, leveraging Proactive Data Containers (PDC) as a tuning technique can significantly enhance performance compared to highly optimized HDF5 implementations [20].

5 Design and Implementation

As discussed in 2, the pre-computation of common mathematical operations in post-processing and analysis applications is a promising approach to improve the performance of HPC workflows without requiring any application changes. More specifically, we demonstrate how moving the computations of the analysis phase to the writing phase can offer multiple benefits.

1) The data is already present in RAM during the writing phase. So, it is the ideal point to compute derived characteristics. For the functions we find to be most suitable, the computational and storage overheads are negligible and therefore do not impact the writing process much. But even if there is an impact, given the read-mostly nature of most data in these systems, speeding up the reading significantly has a bigger impact than marginally slowing down the writing process. 2) By storing the derived characteristics that are much smaller in size they can be stored on the faster and smaller hardware tiers like SSDs or NVRAM. 3) By storing the data in a database we can use optimizations to improve the retrieval performance such as creating indexes on popular columns.

To select suitable computations, we decided to perform a user survey among researchers and study the functions offered by the library very commonly used to analyse climate data, namely the climate data operators (CDOs). By including a subset of the CDOs which can show how analysis libraries can be tied to the I/O engines.

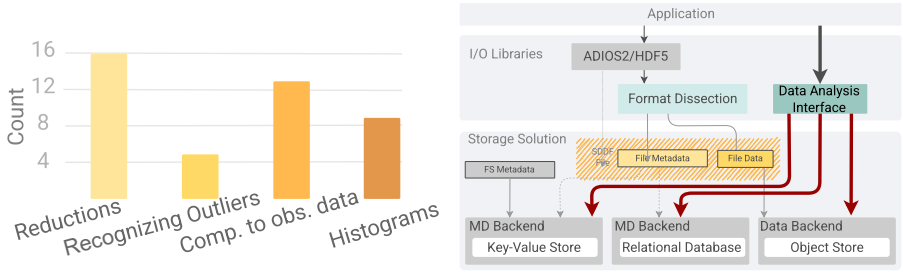
5.1 Survey

The survey investigated the usage of self-describing data formats, e.g. the number of variables and timesteps or the typical hierarchy depth, on the one hand, and interface preferences and used programming languages on the other hand. Furthermore, typical post-processing operations were evaluated as well to inform the pre-computation options. The survey targeted scientists across various institutes, including Sandia National Laboratories (SNL), University of Tennessee, Lawrence Berkeley National Laboratory (LBNL), German Climate Computing Center (DKRZ), Max-Planck-Institute of Meteorology (MPI-M), German Electron Synchrotron (DESY), Center for Bioinformatics (ZBH), University of Hamburg (UHH), Helmut-Schmidt University (HSU), German Aerospace Center (DLR), DDN, and McGill University. In addition to direct outreach, the survey was shared on the HDF5 user forum and the ADIOS discussions on GitHub,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

facilitating valuable input from library developers. The survey engaged a total of 38 participants, with 22 providing complete responses. Note that it is not intended to be a representative study of one or more scientific domains, it simply serves as a way to make a more informed decision about what operations to chose. Given the paper's focus on the merits of pre-computation, only the results for typical post-processing and analysis operations are included and discussed here for brevity.

The survey results illustrated in Figure 3a show that reductions like computing averages, extrema, sums and variances are most common. Aside from that, researchers often compare simulation data to observational data, create histograms and identify outliers. As the reductions are used across various fields from climate research to astrophysics we decided to focus on them to show how this pre-computation can benefit different domains.



(a) Typical post-processing operations (b) Data Analysis Interface (DAI)

Fig. 3: Survey results and the architecture of the DAI

5.2 Climate Data Operators

The climate data operators (CDO) are a command line tool offering various operations on climate data [18]. These range from file operations such as copying or splitting to arithmetic functions. Besides, modifications such as setting the time or the grid info are also numerous. The arithmetic functions contain operations to evaluate an expression or compare simulation data to constants. Additionally, CDOs offer the computation of correlation and covariance in the grid space or over time, as well as regressions to find trends in the data. Interpolation and remapping, as well as spectral and wind transformation, are common as well. We chose to focus on operations with a low overhead in terms of computation and storage space. Reductions like the sum and mean are ideal candidates as they reduce a potentially large dataset to a single result without requiring complex calculations. *Climate Indices* To highlight how the option for pre-computation can also help more specific scenarios and not just generic derived data characteristics, the DAI offers functions to trigger the automatic computation of climate indices. They are part of the European Climate Assessment (ECA) and are used

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

to evaluate temperature and precipitation extremes and their variability. The indices are computed for a time period, typically years. An example of a climate index is the frost day index which counts the number of days per year that have a minimum temperature below 0°C.

5.3 Data Analysis Interface

Given the results of the user survey and the analysis of CDO functionality, we chose to support the following functions: various selection operations for data retrieval, statistics like the mean, the sum and the variance as well as the climate indices.

Retrieval Our choice of selection operators for the DAI is motivated by the 6 typical access patterns for a variable in a self-describing data format [24]. A variable can be read as the whole domain, an xz-plane, xy-plane, yz-plane, a sub-area or a partial area of a plane. Thus, the DAI supports the selection of single database fields, timesteps or a subset of a variable. The statistics offered by the CDOs contain various ranges, e.g. values over an entire ensemble, a field, and a time range. Thus, the DAI supports different granularity levels, namely the block level, step level or the variable level.

Pre-computation Most functions of the DAI are related to specifying pre-computations and respective settings. Only statistics that are implemented manually in the engine can be precomputed. There is currently no option to define arbitrary mathematical operations as the support is more of a technical challenge in this setup than a research question. So, we focused on showing that offering these functions even in the simplest way can greatly benefit the analysis performance. For the same reasons, we assumed that the data uses a structured and uniform grid as this makes the calculation straightforward. Otherwise, a suitable interpolation must be used which could also involve remapping the data.

Tagging We also offer to tag specific features for a variable, that fulfill a condition specified by the user. For example, all blocks for the temperature variable could be tagged that have a maximum temperature above 25°C. This allows to mark and later retrieve interesting data quickly. In the future, this tagging can be done completely in the configuration file to provide optimal flexibility.

Caching Our current implementation caches previous block and step results, leading to increased memory consumption. Transitioning to calculations that rely solely on the latest result could mitigate this issue. Furthermore, keeping data consistent is not a problem as ADIOS2 does not offer updating, overwriting or deleting data for a variable. This type of data manipulation is handled at the file system level instead.

Convenience and Defaults To make the lives of application developers and users doing post-processing easier, several functions are computed automatically by default. For one, the mean value, the sum, and the variance are computed for every block alongside the extrema that are already present in the BP formats. So, to enable the computation of these statistics, setting the engine type to `DB-engine` is sufficient meaning that the application does not need to be recompiled. If the pre-computation is not sensible for a specific scenario, the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

user can specify to use only the metadata offered by the native BP formats. One reason to opt out of the default to automatically compute the additional statistics may be the concern of the increased data size. For every block[5], three additional variables of the same data type as the ADIOS2 variable are stored in the database. Assuming each array has at least one element or character and the variable is not a single value, the minimal size is 143 bytes. Assuming strings with a length of 9 characters and a three-dimensional dataset, the size is already 247 bytes. This example shows that the project namespace, the file name and the variable name can contribute significantly more to the entry size than the additional statistics. Given the small storage overhead, we find storing three additional doubles to be justifiable. Especially when considering that this means the user does not have to change anything in the application and can still benefit from the pre-computation. Due to the expected performance improvement when accessing the data, the pre-computation of the three statistics is the default. To access the pre-computed values through the CDO interface for the post-processing, writing a small wrapper is sufficient to map the CDO calls to the respective JULEA DAI calls.

6 Evaluation

In the following, we present the evaluation of the JULEA engines and the DAI. The query application representing the analysis application is run on one of the `smallerNode`. The JULEA servers run on the `largeNode`.
`smallerNode` is equipped with an AMD Epyc 7443 CPU (2.85 GHz, 24 cores), 128 GB of RAM, and a Western Digital 500 GB WD5000AAKS HDD, capable of 126 MB/s sustained throughput.
`largeNode` is equipped with an AMD Epyc 7543 CPU (2.8 GHz, 32 cores), 1024 GB of RAM, and an Intel SSDSC2KB960G8 SSD (960 GB) with a max. performance as per specification of 510 MB/s (write), 560 MB/s (read).
FS The local FS on the cluster is ext4 with a block size of 4096 bytes. Ceph is used as the distributed FS holding, for example, the home directories. The BP engines write to Ceph, whereas the JULEA engines write to the local storage of the nodes running the JULEA servers. This setup is chosen to avoid the overhead of an additional distributed FS when using JULEA.

Software As the JULEA benchmark revealed, LevelDB has the best overall performance for different operations, including writing and reading. MariaDB is the best available option in JULEA to store data in a database. While the performance of SQLite running in RAM is better, the data are not persisted. Therefore, MariaDB was chosen. MariaDB is run in a Singularity container that uses the current docker image. Furthermore, we use ADIOS 2.7.

Real-world application To evaluate write and read performance, we utilized the HeatTransfer application [6] from the ADIOS2 repository. It solves the 2D

[5] A block is the data written by one MPI process in one step.
[6] https://github.com/parcio/adios2/tree/master/examples/heatTransfer

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

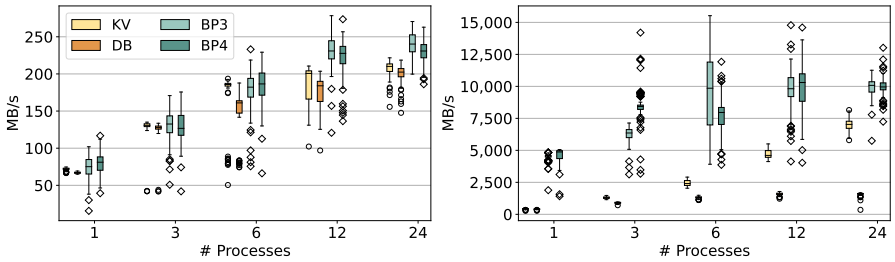
Poisson equation using finite differences for the temperature distribution in homogeneous media. A constant matrix size of 1024^2 was used. We introduced a second variable, precipitation, to facilitate more complex DAI queries. Furthermore, we increased the simulation steps to 100 rather than the matrix size, intentionally stressing the JULEA engines with additional metadata to rigorously test their capacity under more demanding conditions.
ADIOS2-Query and DAI-Query Two applications were developed to assess the DAI. Both operate within a single process, as post-processing tasks like visualization or plotting scripts are often not parallelized. The first application, ADIOS2-Query, utilizes the ADIOS2 interface for data access, illustrating how the native library handles posed queries. Notably, the ADIOS2 interface cannot access the new custom metadata. The second application, DAI-Query, employs the DAI interface to query the database and custom metadata. Pre-computation time is not separately considered, as it was previously measured in the write performance of the JULEA engines.
Raw Data All evaluation results, as well as the specifics of how we compiled and set up the system, will be published along the survey results [7].

6.1 Write and Read Performance

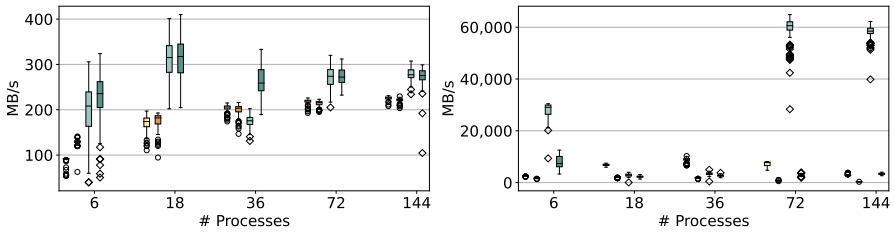
In the following, the I/O performance of the two ADIOS2 (BP3 and BP4) and the two JULEA engines (KV-engine and DB-engine) is measured to demonstrate the applicability of our approach across the entire application life cycle not just the analysis phase. The results shown in the following figures are the mean write and read throughput averaged over all processes performing I/O in a step for a total of 100 steps. The write and read throughput for one and six nodes are shown in Figure 4 for BP3, BP4 and both JULEA engines. The results for 2 and 4 nodes offer no additional insight and are omitted for the sake of space. For both JULEA engines, the performance gain grows less with a higher number of processes but does not decrease. In most cases, the engine using the key-value store is only slightly better than the one using the relational database. Given the performance differences observed benchmarking the specific backends, this is surprising. We measured the performance for `put`, `get` and `delete` operations both unbatched and batched[8]. While LevelDB reached between 59,000 and 65,000 operations per second, MariaDB only achieved between 1,000 to 9,000 operations per second. Note the wide variation, especially for the BP engines. For reading, the performance ranges from 4 GB/s to about 15 GB/s when using BP3 on 6 processes on one node as shown in Figure 4a. This is due to caching effects that we could not eliminate. Overall, the evaluation shows that our engines provide a comparable write performance while the read performance is lacking due to caching mechanisms on ADIOS2's side.

[7] https://github.com/kiraduwe/julea
[8] The plots for these results are not included for the sake of space

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8



(a) Write (left) and read (right) for 1 node



(b) Write (left) and read (right) for 6 nodes

Fig. 4: HeatTransfer Results: Write and read performance for 1 node (a) and for 6 nodes (b). ADIOS2 version 2.7.1 was used.

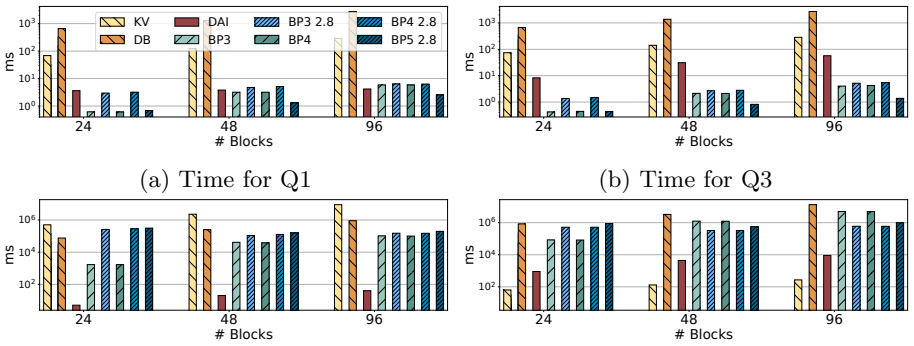
6.2 Query Time

To evaluate the performance of the DAI and the pre-computed statistics we evaluated several queries. Four queries have been derived from the use cases we examined. They have been phrased such that they represent a real-world question. Our engines have been developed using ADIOS2 version 2.7. In order to also compare the DAI to the newer BP5 format, we included measurements for ADIOS2 version 2.8 as noted in Figure 5.

Query 1: Find all locations where the temperature is between -42°C and 42°C. The first query operates on file metadata in the original BP formats. All engines, first retrieve all blocks where the temperature variable meets the condition and then the corresponding x and y coordinates are checked to deduct the location. The performance of the JULEA engines, when employing the ADIOS2 interface, is expected to be, at best, comparable to that of the BP engines, in part because they have an index for this type of metadata. This could be added to the JULEA engines as well to improve their performance in the future.

Figure 5a illustrates the results, indicating that the runtime for the JULEA engines, influenced by backend technology performance, is longest with MariaDB, reaching 2,770 ms for 96 blocks. Efforts to enhance performance involve running the database server on faster hardware, ideally in NVRAM, although this is currently unavailable in the utilized cluster. When the DAI (DAI-Query) is used instead of the ADIOS2 interface (ADIOS2-Query), the performance is improved significantly from 2,770 ms to 4 ms. This shows the advantage when the interface can access the custom data.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8



(a) Time for Q1

(b) Time for Q3

(c) Time for Q2

(d) Time for Q4

Fig. 5: The runtime for Q1 (a), Q3 (b), Q2 (c) and Q4 (d). The results were written with 1, 2 and 4 nodes with 24 processes per node which equal writing 24, 48 and 96 blocks. Note the logarithmic scale on the y-axis and the value ranges. Version 2.7 (teal) and 2.8 (blue) of ADIOS2 are used.

Query 2: What is the highest mean value of a location?
This query showcases the efficiency of pre-computed custom metadata, focusing on mean value calculations. ADIOS2-Query requires reading the variable data to compute the mean value, introducing additional time, while DAI-Query benefits from pre-computed mean values during variable writing. To optimize, additional metadata computations are combined where possible, such as reusing the sum for mean and standard deviation calculations. DAI-Query outperforms ADIOS2-Query for DB-engine, with runtimes of 41 ms 96 blocks, compared to 899,400 ms, a factor of 21,940. Notably, the DAI interface achieves impressive results even with MariaDB as the underlying technology.

Query 3: Which location had the largest increase in maximum temperature between step 1 and 100?
The third query assesses metadata querying performance across multiple steps, specifically comparing maximum temperatures for all blocks in steps 1 and 100. Unlike prior queries, this scenario avoids reliance on pre-computed features and challenges engines by requiring the reading and comparison of maximum temperatures across steps. Figure 5b illustrates consistent runtimes for JULEA engines, regardless of using one or two servers. The DB-engine engine has the longest runtime which is in part because of the lower performance of MariaDB as mentioned before. Notably, using DAI with JULEA clients achieves significantly shorter runtimes (around 31 ms for 96 blocks) compared to using MariaDB through the ADIOS2 interface (2,720 ms), even outperforming LevelDB. The BP engines have a small runtime, for example, 5 ms for BP3. A possible reason is the number of small reads that have to be performed using the JULEA engines. Currently, there is no way to read all file metadata for a specific variable at once. In contrast, the BP engines only need to read the metadata file of their formats and all blocks' maxima are available to the application. One option to mitigate

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

the drawbacks would be to offer an iterator to the application to allow reading multiple entries more efficiently.

Query 4: What was the highest precipitation sum for a location when the maximum temperature was above 40°C?
The chosen query for the final evaluation involves multiple variables, adding complexity compared to previous queries. In Figure 5d, the KV-engine engine outperforms others with a runtime of approximately 270 ms. In contrast, DB-engine exhibits a longer runtime of 9060 ms for 96 blocks, a factor of about 35.

	BP	**J-KV**	**J-DB**	**DAI**
Query 1	1	2	3	1
Query 2	2	4	3	1
Query 3	1	3	4	2
Query 4	3	1	4	2

Table 1: Assessment of the query runtimes from best (1) to worst (4).

6.3 Discussion

When bringing the format dissection, the file metadata management in a relational database and the DAI together, they can compete with the BP engines when writing data. However, they shine when the custom metadata is relevant to the posed question and even outperform the native engines significantly by several orders of magnitude .

Query results Some interesting aspects become more evident in the condensed query results in Table 1. First, the query selection was varied and focused on different capabilities. Second, no candidate is best for each scenario. This is not surprising since previous work on access patterns for variables in self-describing data formats indicated that they could be very varied [24]. Third, the format dissection itself is not beneficial when a slow backend is used, as evidenced by the subpar results of DB-engine, whereas using a key-value store proved more efficient, largely due to the superior performance of LevelDB over MariaDB. Fourth, the pre-computation of statistics combined with the DAI is a great addition and helps improve the performance significantly even if a slow backend like MariaDB is used.

Querying across files Querying across files was not assessed due to the cluster's limited storage capacity. Despite this, the centralization of file metadata does allow for quicker cross-file queries, avoiding the need to open multiple files.

Data modelling One major challenge is that data can be modelled differently in different formats. For example, in HDF5, time can be represented using a specific array dimension, or multiple datasets can be written, each corresponding to a specific step, or even individual files for each step. This complicates the computation of additional data characteristics, e.g. for a time step. Furthermore, different scientific domains, such as climate simulations versus particle physics,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

introduce varied requirements for statistical analyses and grid structures.
Automatic detection of new statistics To automate the detection of statistics to precompute the following approaches could be used: 1. Machine Learning Models: Implement algorithms to predict useful statistics based on past data access patterns during analysis, enabling adaptive adjustments over time. 2. Metadata Analysis: Leverage insights from metadata such as access frequency, data size, and type to inform precomputation decisions. 3. Feedback Loops: Incorporate direct user input to tailor precomputation strategies to actual user needs. Exploring these approaches allows storage engines to dynamically adapt and optimize performance to support diverse analytical requirements of HPC data.

7 Conclusion

We present a novel data management technique utilizing ADIOS2 to boost query performance in HPC analysis applications by automating the precomputation of frequently used data characteristics that were identified in a user survey. Our Data Analysis Interface (DAI), integrated within the JULEA storage framework, demonstrates performance improvements by a factor of up to 22,000 —while supporting domain-specific features, thereby showcasing its adaptability and effectiveness across various scientific fields.

References

1. Brandt, S., Maltzahn, C., Polyzotis, N., Tan, W.C.: Fusing data management services with file systems. In: Proceedings of the 4th Annual Workshop on Petascale Data Storage. p. 42–46 (2009), https://doi.org/10.1145/1713072.1713085 5
2. Breitenfeld, M.S., Pourmal, E., Byna, S., Koziol, Q.: Achieving High Performance I/O with HDF5. http://tinyurl.com/hdf5tutorial (Feb 2020), acc: 28.08.2022 1
3. Byna, S., Breitenfeld, M.S., Dong, B., Koziol, Q., Pourmal, E., Robinson, D., Soumagne, J., Tang, H., Vishwanath, V., Warren, R.: Exahdf5: Delivering efficient parallel I/O on exascale computing systems. J. Comput. Sci. Technol. **35**(1), 145–160 (2020), https://doi.org/10.1007/s11390-020-9822-9 5
4. Duwe, K., Kuhn, M.: Dissecting self-describing data formats to enable advanced querying of file metadata. In: SYSTOR. pp. 12:1–12:7. ACM (2021). https://doi.org/10.1145/3456727.3463778 4
5. Duwe, K., Lüttgau, J., Mania, G., Squar, J., Fuchs, A., Kuhn, M., Betke, E., Ludwig, T.: State of the Art and Future Trends in Data Reduction for High-Performance Computing. Supercomput. Front. Innov. **7**(1), 4–36 (2020) 1
6. Grawinkel, M., Nagel, L., Mäsker, M., Padua, F., Brinkmann, A., Sorth, L.: Analysis of the ECMWF storage landscape. In: FAST. pp. 15–27. USENIX Association (2015) 5
7. Gray, J., Liu, D.T., Nieto-Santisteban, M.A., Szalay, A.S., DeWitt, D.J., Heber, G.: Scientific data management in the coming decade. SIGMOD Rec. **34**(4), 34–41 (2005) 2
8. Isakov, M., Rosario, E.D., Madireddy, S., Balaprakash, P., Carns, P.H., Ross, R.B., Kinsy, M.A.: HPC I/O throughput bottleneck analysis with explainable local models. In: SC. p. 33. IEEE/ACM (2020) 1

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8

9. Korenblum, D., Rubin, D.L., Napel, S., Rodriguez, C., Beaulieu, C.F.: Managing biomedical image metadata for search and retrieval of similar images. J. Digit. Imaging **24**(4), 739–748 (2011), https://doi.org/10.1007/s10278-010-9328-z 5
10. Kuhn, M., Duwe, K.: Coupling Storage Systems and Self-Describing Data Formats for Global Metadata Management. In: 2020 CSCI. pp. 1224–1230 (2020). https://doi.org/10.1109/CSCI51800.2020.00229 4
11. Lawson, M., Gropp, W., Lofstead, J.F.: EMPRESS: accelerating scientific discovery through descriptive metadata management. ACM Trans. Storage **18**(4), 34:1–34:49 (2022) 4, 5
12. Lofstead, J.F., Jimenez, I., Maltzahn, C., Koziol, Q., Bent, J., Barton, E.: DAOS and friends: a proposal for an exascale storage system. In: SC. pp. 585–596. IEEE Computer Society (2016) 5
13. Lofstead, J.F., Zheng, F., Klasky, S., Schwan, K.: Adaptable, metadata rich IO methods for portable high performance IO. In: 23rd IEEE IPDPS. pp. 1–10. IEEE (2009) 1
14. Mu, J., Soumagne, J., Byna, S., Koziol, Q., Tang, H., Warren, R.: Interfacing HDF5 with a scalable object-centric storage system on hierarchical storage. Concurr. Comput. Pract. Exp. **32**(20) (2020) 6
15. Niazi, S., Ismail, M., Haridi, S., Dowling, J., Grohsschmiedt, S., Ronström, M.: Hopsfs: Scaling hierarchical file system metadata using newsql databases. In: FAST. pp. 89–104. USENIX Association (2017) 5
16. Purandare, D., Bittman, D., Miller, E.: Analysis and workload characterization of the CERN EOS storage system. In: CHEOPS@EuroSys. pp. 1–7. ACM (2022) 3
17. Rew, R., Davis, G., Emmerson, S., Davies, H.: NetCDF User's Guide - An Interface for Data Access Version 2.4. http://www-c4.ucsd.edu/netCDF/netcdf-guide/guide_toc.html (February 1996), last accessed: 15.07.2022 1
18. Schulzweida, U.: Cdo user guide (Oct 2022), https://doi.org/10.5281/zenodo.7112925 4, 7
19. Settlemyer, B.W., Amvrosiadis, G., Carns, P.H., Ross, R.B., Mohror, K., Shalf, J.M.: It's time to talk about HPC storage: Perspectives on the past and future. Comput. Sci. Eng. **23**(6), 63–68 (2021) 3
20. Tang, H., Byna, S., Bailey, S., Lukic, Z., Liu, J., Koziol, Q., Dong, B.: Tuning object-centric data management systems for large scale scientific applications. In: HiPC. pp. 103–112. IEEE (2019) 6
21. Tang, H., Byna, S., Dong, B., Liu, J., Koziol, Q.: Someta: Scalable object-centric metadata management for high performance computing. In: CLUSTER. pp. 359–369. IEEE Computer Society (2017) 5
22. Tull, C.E., Essiari, A., Gunter, D., Li, X.S., Patton, S.J., Ramakrishnan, L.: The SPOT Suite project. http://spot.nersc.gov/. (September 2013), acc: 2020-10-09 5
23. Uselton, A., Howison, M., Wright, N.J., Skinner, D., Keen, N., Shalf, J., Karavanic, K.L., Oliker, L.: Parallel I/O performance: From events to ensembles. In: IPDPS. pp. 1–11. IEEE (2010) 1
24. Wan, L., Huebl, A., Gu, J., Poeschel, F., Gainaru, A., Wang, R., Chen, J., Liang, X., Ganyushin, D., Munson, T.S., Foster, I.T., Vay, J., Podhorszki, N., Wu, K., Klasky, S.: Improving I/O performance for exascale applications through online data layout reorganization. IEEE Trans. Parallel Distributed Syst. **33**(4), 878–890 (2022) 8, 13
25. Zhang, W., Byna, S., Tang, H., Williams, B., Chen, Y.: MIQS: metadata indexing and querying service for self-describing file formats. In: SC. pp. 5:1–5:24. ACM (2019) 5

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_8