Parallel Computing and I/O

Problem Statement

Many high-performance computing (HPC) applications generate vast amounts of data and write them to parallel distributed file systems where they are kept for further processing. All production-level file systems currently in use offer a POSIX I/O interface that treats file data as an opaque byte stream. As it is not possible to reconstruct the data format from this byte stream without p knowledge, self-describing data formats such as NetCDF (Network Common Data Format) and ADIOS (Adaptable IO System) a widely used to be able to exchange data with other researchers and annotate data with meaningful metadata. For example, the World Data Center for Climate (WDCC) alone holds multiple petabytes (PB) of data in such formats and provides access for oth researchers. The data structure information is encoded in the files themselves, which makes it possible to group and annotate data. Moreover, data can be accessed and interpreted without having knowledge about its structure in advance.

In a typical HPC I/O stack, applications only interface directly with NetCDF, which depends on HDF5 (Hierarchical Data Format) and so on. The coupling between the different layers is loose and mainly used for performance tuning (for example, applications might set optional I/O hints). However, structural information about the data is lost as it is handed down through the layers: Whil an application might pass multi-dimensional matrices of numerical data to NetCDF, MPI-IO is only aware of a stream of element (which can be different simple or composed data types) and Lustre's POSIX interface interprets file data as a stream of bytes.

Additionally, data and metadata are typically separated in such file systems for performance reasons. This leads to two different types of metadata: *file system metadata* (such as ownership and permissions) are stored on the file system's metadata servers, while *file metadata* (such as the type of physical quantities) are stored within files on the file system's data servers. File system metadata is typically in the range of 5% of the overall data volume, which leads to significant amounts of metadata in current mu petabyte file systems.

There are several major problems with the current state:

1. **Weak classification of metadata:** Opening a file and reading specific portions of data (for example, a single output dimension) first requires accessing the file system metadata to identify the servers holding file data; afterwards, file metadata is read to determine which portions of the file have to be accessed; finally, the actual data can be read from the file system. This strict separation between file system metadata and file metadata leads to inefficient file access. Maintaining metadata requires coordinated access to a shared resource and therefore imposes significant synchronization overhead due to strict semantics mandated by POSIX. Moreover, file metadata is distributed across potentially many different data servers that have to be contacted, causing additional overhead. Data servers are commonly optimized for streaming I/O and therefore can not deal well with small random accesses that are typical for metadata requests. They often rely on sequential access patterns to achieve high performance through techniques such as readahead.
2. **Static I/O semantics:** Due to the complex interplay of different components and optimizations in the I/O stack, performance issues are a common occurrence. One of the reasons is the strict POSIX semantics that are typically provided by the underlying parallel file system and forced upon the upper layers. POSIX requires changes to be visibl immediately after a write call and to be performed in an atomic way; this can lead to performance issues in a distribute system due to synchronization overhead. HPC applications often do not require the strict consistency and coherence semantics offered by POSIX but are not able to opt out. Even though higher layers such as MPI-IO typically offer relaxed semantics, their dependence on the parallel file system's POSIX interface effectively negates these relaxation Static semantics is unable to satisfy the requirements of vastly different applications, requiring application-specific workarounds.
3. **Loss of structural information:** To suffice the wide range of requirements which are imposed on an HPC system, a hierarchical structuring of different hardware is used. While the hardware itself is available, new approaches how to us it efficiently need to be developed. In order to gain the most of this layering, the selection of the appropriate technolog for a certain piece of data has to be done carefully; otherwise, data has to be moved across storage tiers repeatedly, which is an expensive operation. Currently, data placement decisions often have to be based on heuristics derived fr

access frequencies and related I/O patterns. Due to the strict separation of layers and the loss of structural informatio through the I/O stack, efficient policies are currently not viable. Furthermore, support for heterogeneous tiers consistir of different hardware types is not advanced enough and therefore the system's capabilities cannot be fully exploited.

Finally, existing file systems, especially parallel distributed file systems, do not provide possibilities for flexible adaptations to the core concepts such as the semantics or file system structure. This makes the evaluation of new propositions very expensive anc ultimately hinders innovation. Another drawback is the fact that a considerable number of file systems is only realized in the kerr space, increasing the complexity of changes significantly. Kernel file systems can typically only be set up and mounted by privileged users. Moreover, integration into the kernel requires a more sophisticated implementation and involves considerable maintenance costs.

The aim of CoSEMoS is to propose and implement significant improvements regarding performance and data management through a novel architecture making use of established technologies from the fields of databases and HPC. By introducing a unified global metadata management combined with adaptable I/O semantics and an intelligent storage selection, the problems discussed above will be addressed. We expect this to have considerable performance benefits as our previous studies have shown that especially metadata performance can be improved by up to two orders of magnitude using similar techniques. Furthermore, by leveraging existing I/O interfaces, this approach is completely transparent for users. CoSEMoS will also provide data analysis interface for efficient post-processing and an export functionality so that files can be exchanged between scientists as before.