Adaptive multi-tier intelligent data manager for Exascale

Jesus Carretero
Javier Garcia-Blas
(jcarrete,fjblas)@inf.uc3m.es
University Carlos III of Madrid
Leganes, Madrid, Spain

Marco Aldinucci
marco.aldinucci@unito.it
University of Torino & CINI
HPC-KTT lab.
Torino, Italy

Jean Baptiste Besnard
jbbesnard@paratools.fr
ParaTools SAS
Bruyères-le-Châtel, France

Jean-Thomas Acquaviva
jtacquaviva@ddn.com
DataDirect Networks Storage
Paris, France

André Brinkmann
Marc-André Vef
(brinkman,vef)@uni-mainz.de
Johannes Gutenberg University
Mainz, Germany

Emmanuel Jeannot
emmanuel.jeannot@inria.fr
INRIA
Bordeaux, France

Alberto Miranda
Ramon Nou
alberto.miranda@bsc.es
ramon.nou@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

Morris Riedel
m.riedel@fz-juelich.de
Juelich Supercomputing Center
Jülich, Germany

Massimo Torquati
massimo.torquati@unipi.it
University of Pisa & CINI HPC-KTT
lab.
Pisa, Italy

Felix Wolf
felix.wolf@tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

ABSTRACT

The main objective of the ADMIRE project[1] is the creation of an active I/O stack that dynamically adjusts computation and storage requirements through intelligent global coordination, the elasticity of computation and I/O, and the scheduling of storage resources along all levels of the storage hierarchy, while offering quality-of-service (QoS), energy efficiency, and resilience for accessing extremely large data sets in very heterogeneous computing and storage environments. We have developed a framework prototype that is able to dynamically adjust computation and storage requirements through intelligent global coordination, separated control, and data paths, the malleability of computation and I/O, the scheduling of storage resources along all levels of the storage hierarchy, and scalable monitoring techniques. The leading idea in ADMIRE is to co-design applications with ad-hoc storage systems that can be deployed with the application and adapt their computing and I/O behaviour on runtime, using malleability techniques, to increase the performance of applications and the throughput of the applications.

[1] Web: https://www.admire-eurohpc.eu/

CF '23, May 9–11, 2023, Bologna, Italy
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0140-5/23/05…$15.00
https://doi.org/10.1145/3587135.3592174

CCS CONCEPTS

• **Software and its engineering → Development frameworks and environments**; **Massively parallel systems**; • **Computer systems organization → Multiple instruction, multiple data**.

KEYWORDS

High-Performance I/O, Malleability, , Monitoring, Scheduling algorithms, Predictive models

ACM Reference Format:
Jesus Carretero, Javier Garcia-Blas, Marco Aldinucci, Jean Baptiste Besnard, Jean-Thomas Acquaviva, André Brinkmann, Marc-André Vef, Emmanuel Jeannot, Alberto Miranda, Ramon Nou, Morris Riedel, Massimo Torquati, and Felix Wolf. 2023. Adaptive multi-tier intelligent data manager for Exascale. In *20th ACM International Conference on Computing Frontiers (CF '23), May 9–11, 2023, Bologna, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3587135.3592174

1 INTRODUCTION

The growing need to process vast data sets is one of the main drivers for building Exascale HPC systems today. However, the flat storage hierarchies found in classic HPC architectures no longer satisfy the performance requirements of data-processing applications. Uncoordinated file access in combination with limited bandwidth makes the centralized back-end parallel file system a severe bottleneck. At the same time, emerging multi-tier storage hierarchies can potentially remove this barrier. But maximizing performance still requires careful control to avoid congestion and balance computational with storage performance. Unfortunately, appropriate interfaces and policies for managing such an enhanced I/O stack are still lacking.

The main objective of the ADMIRE project is to establish this control by creating an adaptive I/O stack that dynamically adjusts computation and storage requirements through intelligent global coordination, the malleability of computation and I/O, and the scheduling of storage resources along all levels of the storage hierarchy. To achieve this, we have developed a software-defined framework based on the principles of scalable monitoring and control, separated control and data paths, and the orchestration of critical system components and applications through embedded control points. The leading idea is to co-design applications with ad-hoc storage systems that can be deployed with the application and adapt the computing and I/O behaviour at runtime by using malleability techniques. Our software-only solution will allow the throughput of HPC systems and the performance of individual applications to be substantially increased – and consequently, energy consumption to be decreased – by taking advantage of fast and power-efficient node-local storage tiers using novel, European ad-hoc storage systems and in-transit/in-situ processing facilities. Furthermore, our enhanced I/O stack will offer quality-of-service (QoS) and resilience.

The ADMIRE project consortium includes 14 partners, some of them are major supercomputing centers such as BSC, JSC, CINECA, KTH, and PNSC.

2 ADMIRE FRAMEWORK ARCHITECTURE

Figure 1 shows the primary components of the framework and the use cases that we will co-design to assess the feasibility of our solutions. It also shows how the information, data, and controls, are exchanged between the components. As may be seen, the heart of the framework is the intelligent controller, a distributed control system integrating monitoring data from the system and applications and applying performance models and predictive AI techniques to make decisions to reshape the application and ad-hoc storage systems configuration on runtime. The Data Scheduler is responsible for the deployment and configuration of the Ad-hoc Storage System, the specification of Quality-of-Service metrics, and the implementation of I/O and data scheduling policies. The Malleability Manager determines and suggests malleable actions related to each running application and ad-hoc storage system. These actions may produce the reconfiguration of processes/threads of a specific application or the deployment/removal of one or more instances of the Ad-hoc Storage System to better balance the computation and I/O requirements.

The storage subsystem is represented in the upper part of the figure and consists of the Ad-hoc and Backend Storage Systems. The former is responsible for providing each application with an ad-hoc high-performance storage system tailored to the application's characteristics. The latter one (Backend Storage) represents the parallel file system used by the HPC platform (e.g., Lustre).

The applications that are being executed in the platform are shown in the left part of Figure 1. ADMIRE-enabled applications can provide user-defined application-specific information to the system to aid the identification of I/O patterns and reconfiguration-safe states in which malleability commands can be executed.

Both applications and storage systems are monitored by the Sensing and Profiling component, which is responsible for collecting system-wide performance metrics at the compute node level that will be stored in an internal database. This component is responsible for generating the performance model of each application, which will be used to support the Malleability Manager and Intelligent Controller in finding the most appropriate scheduling policy. A Monitoring Manager leverages the information stored in the database to generate performance models related to (1) each running application, (2) all storage systems (both ad-hoc and backend), and (3) the compute nodes.

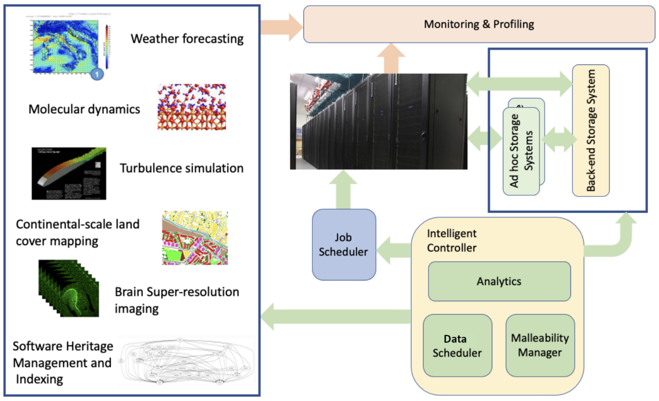

Figure 1: High-level architecture of ADMIRE framework.

3 THE INTELLIGENT CONTROLLER

The heart of ADMIRE is the Intelligent Controller (IC) that allows joining cross-layer information from systems, applications, and users to optimise the throughput of the system and the performance of the applications. The IC communicates with other system components (such as the monitor, applications, job scheduler, etc.) through control points defined for each. It is in charge of coordinating their tasks. The control plane will then steer the main active components.

The IC has different roles. The first one is to collect the current system status using the information collected from Slurm, the Monitoring Manager, the storage systems, and the applications. It includes combined information about the hardware status, the existing running applications, and the storage. The second role of the IC is to generate performance models of these components and use them to predict potential performance bottlenecks in the system. These models will be used to design a schedule solution, which will also consider the suggestions provided by the Malleability Manager and the Data Scheduler. Finally, the other central role of the IC is coordinating the actions taken by other ADMIRE components. Examples of these actions are (1) to activate/deactivate each application monitoring and (2) to send the malleable decisions to the Data Scheduler or the applications in case of I/O-related or application-related decisions, respectively. Figure 2 shows the control flow of applications in the ADMIRE framework and the role of the Intelligent Controller.

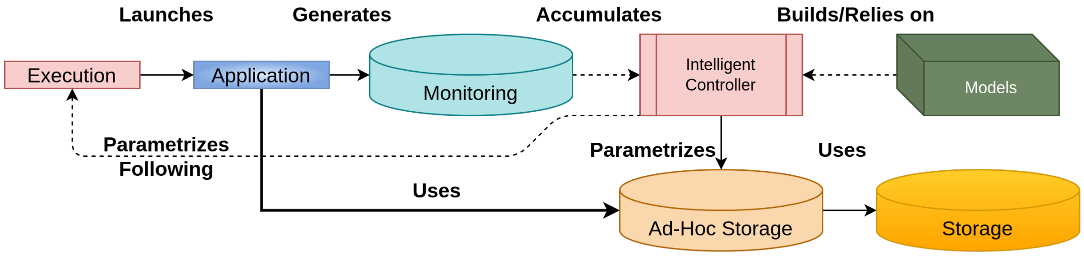

Figure 2: ADMIRE control workflow.

To accomplish the former roles, the IC is designed as a multi-criteria distributed component that integrates cross-layer data to have a holistic view of the system. The IC will also make intelligent analyses to adapt dynamically the system to current and future workloads to increase its throughput. It will enable the global orchestration of the Exascale system components through an intelligent dynamic control plane. In this way, the IC interoperates with the Monitoring Manager, the Applications, the Resource Manager (Slurm), the Malleability Manager, the Data Scheduler, and the data plane through predefined control points and interfaces.

The **analytics** component improves Computing and I/O system behaviour and facilitates anticipatory decisions for resource allocation based on the knowledge collected from the I/O systems, applications, and the batch system. The IC will collect information provided by the monitoring system, which will be analyzed using novel data-centric machine-learning techniques to predict application and system behaviour. It also obtains additional information from the historical job records of previous runs. In this context, the analytics component of the IC will holistically integrate and analyze cross-layer system data to dynamically and intelligently steer the system to tune I/O performance at the system scale. The strategy of this component consists of training model classifiers that can handle jobs with heterogeneous I/O patterns, detect congestion, perform resource provisioning, and perform anomaly detection that will help the IC to identify and predict potential risks and failures of applications.

The **IC scheduler** leverages and combines (1) the performance models generated by the Analytic component, (2) the malleability suggestions provided by the Malleability Manager, and (3) the I/O scheduling solutions obtained from the Data Scheduler component. It generates a new scheduling solution using multi-criteria objective trade-offs such as response time vs throughput vs energy consumption (utilizing machine learning algorithms) and considering the current state of the platform, the processes, and their near future state. The multi-criteria goals will be provided either by the system administrator or by the job when submitted.

The IC also provides **application support** to achieve an intelligent tuning of parallel applications on runtime through the control points. Some services designed are: dynamically enable/disable control domains by activating/deactivating control agents; provide an open API for the implementation of new control algorithms; call the monitoring and profiling service to find out system and applications I/O state information; allowing insertion of processing rules on incoming data such as filters for supporting dynamic in-situ/in-transit processing. Each active component will expose a series of parameters that are to be configured in a control point (e.g., the number of servers to stripe a file, the block size, etc.). The algorithm in each control point will perform a series of actions targeted to drive the system towards the desired goal. These actions include: finding out the system state based on the monitoring service, inserting processing rules on incoming data such as filters, communicating with control agents, and taking reconfiguration decisions.

The Malleability Manager component and the Data Scheduler are presented in more detail in the next sections.

All communications between the IC, ADMIRE components, and applications are implemented in *libicc* with remote procedure calls (RPCs), a paradigm that local calls are executed on remote resources using Mercury and Margo frameworks, both well in use in the HPC community. Mercury is an RPC library specifically designed for high-performance computing (HPC) systems. Margo is a newer C library built on top of Mercury and the Argobots user-level threading framework. Both Mercury and Margo are actively developed under the umbrella of the Mochi project, a collaboration between Argonne National Laboratory, Los Alamos National Laboratory, Carnegie Mellon University, and the HDF Group. These libraries have been chosen for the ADMIRE project because they couple a low overhead with portability and abundant documentation. Both of them are also available under a free license.

4 HOLISTIC MONITORING

Figure 3 shows the architecture of the monitoring facilities developed in ADMIRE. In order to create a feedback loop between the application state and its expression on a parallel machine, it is crucial to deploy a dedicated measurement infrastructure. For this reason, early in the Admire project, a new measurement infrastructure was devised, specifically targeting the malleability use case. Its first peculiarity is being *always-on*, constantly observing how programs run. Consequently, measurements have to be low-overhead and performance data must be carefully chosen to avoid data-management issues. To match these requirements, node-level monitoring is done with the LIMITLESS monitoring tool which features a Tree-Based Overlay Network. Such a network enables the runtime reduction of performance data, tracking a whole machine at a single point, in close to real-time. On the application side, we performed our instrumentation (1) at the level of the programming interfaces (API) and (2) inside the various ad-hoc storage featured in the project. As far as POSIX I/Os are concerned, we relied on *strace* to perform dynamic attach and detach when more verbose data are needed.

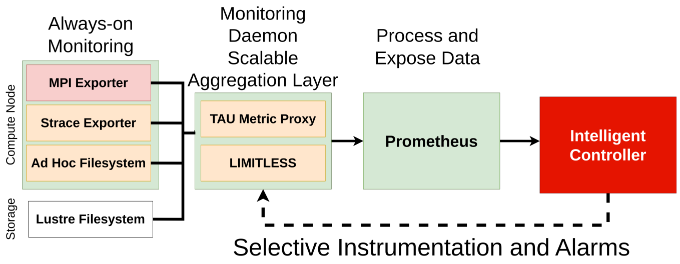

Figure 3: Monitoring framework in ADMIRE.

As malleability is all about the temporal behaviour of a given program, we retained primarily temporal measurements. To do so, we leveraged a proven cloud technology: the Prometheus Time-Series Data-Base (TSDB), that allows to aggregate counters from multiple data sources (*exporters*). These counters are then aggregated over time in the TSDB, which then features a rich REST interface for data consumption. Prometheus periodically pulls data from the various exporters by issuing HTTP requests. As HTTP was not practical to track a large changing set of MPI processes, we developed a specifically tailored aggregating push gateway, relying on UNIX sockets. This push gateway, called the *TAU metric proxy*, acts as a proxy between the data sources and the Prometheus TSDB. Prometheus only polls the proxy and thus a single endpoint. Dealing with the MPI process they however *push* their data in the proxy – simplifying the wiring process.

Due to the verbosity of temporal data, we only generate such *time series* at node level – conversely to per job level. Yet, job-level data are also important, particularly to generate per-job models such as scaling. To this purpose, we implemented a *systematic profiling* engine that stores, for each job, the complete set of afferent counters. Eventually, counters are also reduced thanks to the LIMITLESS TBON, enabling close to real-time tracking of machine-wide values.

5 AD-HOC STORAGE SYSTEMS

Due to the appearance of data-demanding high-performance applications, multiple software solutions have been introduced to cope with challenges along the entire I/O software stack [4], such as high-level I/O libraries, parallel file systems (PFS), and I/O middleware. They aim to reduce challenging I/O accesses to the main storage backend and increase application I/O performance. Such techniques range from general file system optimizations, capturing I/O requests first in an I/O forwarding layer [1] , to employing entirely separate ephemeral file systems that are deployed *ad-hoc* [2]. Ad-hoc file systems have been proposed as a feasible solution as they can exploit local storage resources on the compute nodes and adapt to the needs of a particular application [6, 7].

ADMIRE I/O optimizations are based mostly on using ad-hoc file systems to efficiently use node-local storage technologies, e.g., SSDs or future *non-volatile memories* (NVMs), to reduce the pressure and cross-application interferences on the central PFSs that are used in HPC systems, e.g., Lustre or GPFS. The ad-hoc file systems are further developed and integrated into the ADMIRE ecosystem adding support for fast storage technologies and new data placement algorithms. Resilience mechanisms are added to the ad-hoc file systems so that they can run for longer periods, e.g. in application workflows consisting of multiple consecutive compute jobs that can use the same shared namespace without moving data.

The ad-hoc file systems are transparently used by applications, and they provide connections to the Data Scheduler. In this context, the Data Scheduler manages the ad-hoc file system instances and orchestrates the data movement between them and the PFS. Further, it forwards malleability decisions from the malleability manager to the ad-hoc file system. Therefore, the ad-hoc file systems are also significantly extended in ADMIRE to support malleability and react to an application or HPC system requirements. Examples for *I/O malleability* are expanding and shrinking ad-hoc storage systems, redistributing data to better fit the application access patterns, or entirely changing POSIX requirements and relaxing cache consistency guarantees, for instance. For the latter, it is worth noting that HPC applications generally do not require all POSIX-provided features [8, 9].

Currently, three ad-hoc file systems are provided in ADMIRE: GekkoFS, Hercules, and Expand. All of them are highly scalable distributed file systems for HPC clusters and run entirely in user space. GekkoFS [7] can aggregate the local I/O capacity and performance of compute nodes to produce an ephemeral high-performance storage space for applications. Using GekkoFS, HPC applications can run isolated from each other regarding I/O, which reduces interferences and improves application I/O performance. Further, GekkoFS has been designed with configurability in mind and allows users to fine-tune several of the default POSIX file system semantics, e.g., support for symbolic links, strict bookkeeping of file access timestamps and other metadata, or even modifying entire file system protocols relaxing POSIX semantics. Hercules [6] is a distributed ad-hoc in-memory ephemeral file system, following the client-server approach, that can be deployed with the servers attached to the same nodes of the application or detached in independent nodes. The Hercules API provides a set of deployment methods where the number of servers conforming the instance, as well as their locations, buffer sizes, and their coupled or decoupled nature, can be tuned. To maximize parallelism, Hercules follows a multi-threaded design architecture, provides distributed metadata management, and multiple data location policies that can be tuned at the dataset level for each deployment, avoiding the need for global data mapping and providing benefits such as data locality exploitation and load balancing. The Expand (Expandable Parallel File System) [3] ad-hoc file system combines multiple data servers to create distributed partitions where files are striped using standard protocols, such as MPI or NFS, for the client-server communications, so that it does not need to modify applications for its deployment. Expand can create permanent storage relying on the underlying storage levels and can be connected easily with external NFS or cloud storage services. Hercules and Expand are fully POSIX-compliant file systems.

6 MALLEABILITY AT COMPUTING AND I/O LEVEL

ADMIRE is working towards a solution that combines both computational malleability and I/O malleability, with the goal of achieving a balance between the computation and the I/O requirements of compute jobs. In order to apply malleability at both application and resource management levels (node, OS, and runtime system), we offer applications a programming model that allows them to define a set of so-called *scheduling points*, each representing the possible opportunities for applying malleable operations. At node level, we interact with the Slurm Resource Management System (RMS) to fetch/release resources dynamically, as traditional Slurm assumes all compute jobs to be rigid.

Nevertheless, since this usage of Slurm is often not sufficient for ADMIRE goals, we are also implementing a *Malleability Manager* to suggest scheduling solutions based on a set of malleability mechanisms. Figure 4 shows the interaction of the ADMIRE framework with Slurm in order to support malleable resource allocation. As shown, the IC sends an *expand*/*shrink* order through *libicc* to the FlexADM application manager. The application manager then may request or return resources to Slurm and reorders the application's world to expand/shrink the restructured processes.

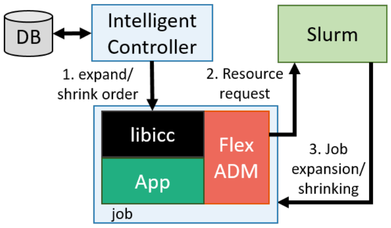

Figure 4: Resource allocation by SLURM.

With this approach, an application can be executed on a dynamically increasing/decreasing number of compute nodes. By means of malleability, it is thus possible to change both an application's CPU performance and I/O bandwidth usage. Algorithm 1 shows the Malleability Manager workflow for the general case of inter-node malleability using Slurm. While this will be the more typical application of the mechanism, we are also considering other options like having a pre-allocated pool of resources and applying intra-node malleability if CPU cores are available.

The *Data Scheduler*'s (DS) goals are supporting ad-hoc storage systems as fundamental pieces of ADMIRE's HPC ecosystem and providing mechanisms to apply malleability strategies in terms of I/O resources. To achieve this, the DS's main responsibility is to coordinate input information from the IC, Slurm, and the MM in order to efficiently orchestrate the movement of *datasets* (i.e., files or objects) between the ad-hoc storage systems and the PFS. It is also responsible for dynamically reconfiguring existing ad-hoc storage instances (e.g., by adding/reducing the number of I/O servers) so as to adapt to I/O malleability decisions from the IC/MM.

As shown in Figure 5, ADMIRE's DS is connected to the control points of user processes indirectly via the IC, which serves to gather information from applications and other ADMIRE components to properly steer data transfers. It is also directly connected to the control points in ad-hoc storage systems in order to deploy and reconfigure them as needed, and to provide them with malleability solutions coming from the IC/MM tandem. The DS also offers a MPI-based *Data Mover* sub-component to ADMIRE, that is responsible for actually copying data in parallel. Similarly to applications, the goal is for its ranks to be flexibly adapted to the system conditions, so that it can transfer data as efficiently as required by the IC.

Algorithm 1 Malleability Manager workflow for inter-node malleability using Slurm.

```
The Malleability Manager is executed
for each running application do
    Slurm: job execution
    Intelligent Controller: system status notification including application performance
    Malleability Manager: application performance analysis
    Malleability Manager: malleable action
    Intelligent Controller: malleable action analysis
    if malleable action is performed then
        Intelligent Controller: for malleable expansion, allocate new resources via Slurm
        Intelligent Controller: wait for the application
        Intelligent Controller: malleable action forwarding
        Intelligent Controller: wait for completion
        Intelligent Controller: for malleable shrinking, release existing resources via Slurm
        Intelligent Controller: application status update
    end if
end for
```

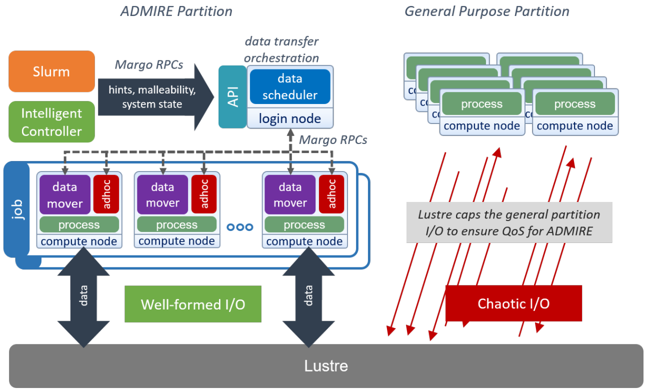

Figure 5: Orchestrating I/O resources in ADMIRE.

Since dataset transfers correspond to actual PFS accesses, properly orchestrating them is crucial to minimize congestion and maximize I/O performance. This is challenging because: 1) ad-hoc storage space is limited, which means that any scheduling algorithm controlling the transfers should strive to keep enough free space to ensure applications do not run out of space; 2) applications should be able to progress as long as they have the data they need, i.e., data should be staged in concurrently with application computations and should ideally be available just before an application needs to access it; and 3) data that is meant to be persistent should be staged out to the PFS as soon as it is no longer needed.

Thus, for data staging to happen *on time*, the DS leverages the application I/O models provided by the IC to predict the frequency and volumes of an application I/O phases, extrapolating them as needed to different node and process configurations. Moreover, since the ADMIRE framework is capable of determining the datasets used by applications (either via user hints or by the extrapolation of historical profiles), the DS can precisely estimate the volumes of data that need to be transferred between tiers. This, coupled with the Data Mover I/O patterns corresponding to large sequential transactions, eases the estimation of the bandwidth and target deadlines.

Note that despite the DS's best I/O prediction and orchestration efforts, it is still possible for applications operating outside the ADMIRE framework to overload the PFS so as to render it unusable for certain periods of time. In order to prevent this to a certain extent, and since it's currently not possible to force all applications in the system to use the ADMIRE services, the DS will steer Lustre's QoS/SLA facilities in order to separate *ADMIRE-class* applications from *General Purpose-class* applications. This will allow non-conforming applications to continue working as usual, while placing restrictions on how negatively their I/O patterns might affect well-behaved ADMIRE applications.

Finally, to further steer applications away from the PFS as much as possible, ADMIRE's DS also provides APIs to facilitate the execution of *in-situ* and *in-transit* operations on data, such as visualisation, analysis and transformation.

7 USE CASES CO-DESIGN

Application co-design is an iterative, developmental methodology, whose origin can be traced back to the field of embedded systems. In the context of ADMIRE, the iterative aspect of the co-design process is shown in Figure 6. It is initiated and sustained through the documentation of application requirements. Once this has been done, the characterisation of applications is obtained via profiling. To this end, tools such as Darshan and TAU are utilized. Through profiling, it is possible to gain an in-depth understanding of each of the six application's I/O and compute usage statistics, either on a cumulative scale or by analysing the consumption of resources on a function-level scale to find and mitigate bottlenecks. Post-profiling analysis helps to identify parts of the application which are to be exposed for integration with the ADMIRE framework. The integrated version is profiled yet again to validate the projected improvements and, also, to identify new performance-related issues. This cycle can be repeated any number of times, till the desired level of porting is achieved.

The current use cases are targeted in the co-design process of ADMIRE: *Environment (A1 - ENV),* marine and weather forecasts and simulations; *Molecule Simulation - Quantum-Espresso (A2 - QE),* describes complex electronic interactions; *Turbulence Simulation - Nek5000 (A3 - NEK),* simulations of large-scale turbulent flows; **Remote sensing (A4 - RS),** analysis of remotely sensed images; *Life Sciences - SRRF (A5 - SRRF),* live-cell Super-Resolution Microscopy; and *Software Heritage Analytics (A6 - SHA),* software stack that enables Data Analytics on Software Heritage.

As an example, the Environment Study application [5] WaComM++ kernel has been co-designed by exploiting ADMIRE technologies to apply malleability to increase the number of processors each simulated hour, targeting an almost constant number of computed particles per unit of time.

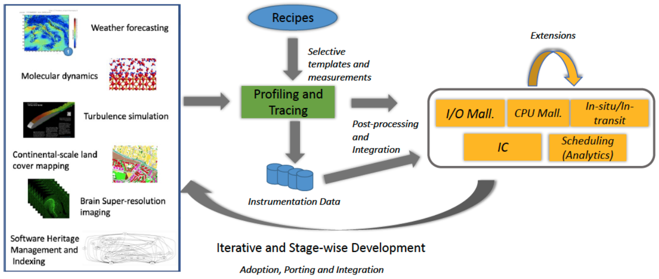
Weather forecasting
Molecular dynamics
Turbulence simulation
Continental-scale land cover mapping
Brain Super-resolution imaging
Software Heritage Management and Indexing
Recipes
Selective templates and measurements
Profiling and Tracing
Instrumentation Data
Post-processing and Integration
Extensions
I/O Mall.
CPU Mall.
In-situ/In-transit
IC
Scheduling (Analytics)
Iterative and Stage-wise Development
Adoption, Porting and Integration

Figure 6: Application co-design iterative life cycle.

8 CONCLUSION

The ADMIRE project is currently ongoing, but it has already contributed with the three ad-hoc files systems developed for HPC environments, a monitoring framework to capture system and data information, a performance model of HPC applications, an API for malleable applications co-design, runtime tools and methodologies for co-design of HPC applications with the Admire framework.

ACKNOWLEDGMENT

This work was partially supported by the EuroHPC project "Adaptive multi-tier intelligent data manager for Exascale" under grant 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1, and by the Agencia Española de Investigación under Grant PCI2021-121966.

REFERENCES

[1] Jean Luca Bez, Alberto Miranda, Ramon Nou, Francieli Zanon Boito, Toni Cortes, and Philippe O. A. Navaux. 2021. Arbitration Policies for On-Demand User-Level I/O Forwarding on HPC Platforms. In *35th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2021, Portland, OR, USA, May 17-21, 2021.* IEEE, 577–586.

[2] André Brinkmann, Kathryn Mohror, Weikuan Yu, Philip H. Carns, Toni Cortes, Scott Klasky, Alberto Miranda, Franz-Josef Pfreundt, Robert B. Ross, and Marc-Andre Vef. 2020. Ad Hoc File Systems for High-Performance Computing. *J. Comput. Sci. Technol.* 35, 1 (2020), 4–26.

[3] Félix Garcia-Carballeira, Alejandro Calderon, Jesus Carretero, Javier Fernandez, and Jose M Perez. 2003. The design of the Expand parallel file system. *The International Journal of High Performance Computing Applications* 17, 1 (2003), 21–37.

[4] Florin Isaila, Javier Garcia-Blas, Jesus Carretero, Rob Ross, and Dries Kimpe. 2017. Making the case for reforming the I/O software stack of extreme-scale systems. *Advances in Engineering Software* 111 (2017), 26 – 31. Advances in High Performance Computing: on the path to Exascale software.

[5] Raffaele Montella, Diana Di Luccio, Pasquale Troiano, Angelo Riccio, Alison Brizius, and Ian Foster. 2016. WaComM: A parallel Water quality Community Model for pollutant transport and dispersion operational predictions. In *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS).* IEEE, 717–724.

[6] Francisco José Rodrigo Duro, Fabrizio Marozzo, Javier García Blas, Jesús Carretero Pérez, Domenico Talia, and Paolo Trunfio. 2015. Evaluating data caching techniques in DMCF workflows using Hercules. (2015).

[7] M. Vef, N. Moti, T. Süß, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann. 2018. GekkoFS - A Temporary Distributed File System for HPC Applications. In *2018 IEEE International Conference on Cluster Computing.* 319–324.

[8] Chen Wang. 2022. Detecting Data Races on Relaxed Systems Using Recorder.

[9] Chen Wang, Kathryn Mohror, and Marc Snir. 2021. File System Semantics Requirements of HPC Applications. In *HPDC '21: The 30th International Symposium on High-Performance Parallel and Distributed Computing, Virtual Event.* ACM.