E10 – Exascale IO

André Brinkmann, Toni Cortes, Hugo Falter, Julian Kunkel, Sai Narasimhamurthy
Johannes Gutenberg University Mainz, Partec and EOFS, Barcelona Supercomputing Center, DKRZ, Seagate

High-performance computing has already turned many compute challenges into storage problems. Checkpointing application state, in-situ analysis of data, or big data applications have significantly increased the pressure on storage systems. Exascale environments will increase this pressure by three additional orders of of magnitude. The E10 – Exascale IO workgroup is an international collaboration between academic organizations and companies that steers the transfer of HPC storage environments into the Exascale era, so that legacy applications and new developments can benefit from new compute and storage architectures. The following white paper describes the E10 approach and architecture and will outline its roadmap.

Introduction

High-performance computing can look back on tremendous progress in the first decade of this century. File systems unfortunately have not been able to keep pace with this progress of compute performance. The average load of storage devices is often very low, in many cases they are only able to use 10% of their potential, while the fraction of time spent in I/O has significantly increased for applications. Scaling to very large numbers of IO nodes has not led to a similar throughput scaling.

Several high-level IO libraries like HDF5, NetCDF, and MPI-IO have tried to address these problems with some success [1][2]. However they are not integrated into the file system and its transactional and recovery semantics. Some of these libraries even build a file system inside a file, thus the real file system cannot leverage its metadata system, ordering constraints, and recovery semantics.

Table 1: *Exascale supercomputer characteristics*[1]

	2012	2020
Nodes	10-100K	100K–1M
Threads/node	10	1000
Total concurrency	100K-1M	100M-1B
Object create	100K/s	100M/s
Memory	1–4PB	30–60 PB
FS Size	10–100 PB	600–3000 PB
MTTI	1–5 Days	6 Hours
Memory Dump	$\leq$ 2000 s	$\leq$ 300 s
Peak I/O BW	1–2 TB/s	100–200 TB/s
Sustained I/O BW	10–200 GB/s	20 TB/s

HPC IO challenges, therefore, have not been resolved so far and the future of Exascale is full of uncertainties (see Table 1 for a prediction). A wilderness of new hardware is expected to arrive – such as deeper hierarchies of storage devices, storage class memory, and large numbers of cores per node. This new hardware may both contribute parts of the solution and also bring new issues to the forefront. This forces storage and application architects to revisit ideas used so far.

E10 as an international collaboration of academic and industrial organizations, working under the umbrella of the Europen Open File System Organization EOFS, tries to focus distributed storage research and implementation efforts into a joint undertaking to increase the probability of a successful adoption of new storage and I/O architectures and technologies

[1] see, e.g, John Bent: Exascale Storage for HPC: Burst buffers with a new Storage API, ISC 2013

for Exascale.

Nevertheless, the past has shown that changes to file system semantics take place at a slow pace. The POSIX interface, while being one of the most serious bottlenecks of scaling parallel file systems, has only been extended slightly over the last 30 years (e.g. with `lstat()`). Facing new challenges, researchers are willing to adapt new interfaces; for example, large-scale data analytics lead to the wide-spread adoption of MapReduce-like environments.

E10 provides a roadmap, which allows to continue the usage of mature file systems, like GPFS and Lustre, over the next years, while providing improvements in the areas of management, interfaces, and predictive analysis. Nevertheless, changes to the underlying storage backend seem inevitable in the long run, so that E10 already prepares new storage semantics and implementations. The aim is to provide a future-proof storage backend, which is able to support next generations metadata and bandwidth performance, while offering file and MapReduce-like interfaces. The foundation are new storage semantics and object storage devices, which support transactional semantics and, therefore, improved parallel access.

This white paper will give an overview about E10 on different architectural granularities and will also provide a short-, mid-, and longterm roadmap of the implementation plans of the different E10 participants.

Exascale Hurdles

The nesting of the HPC I/O stack The I/O stack involves several layers, a typical nesting of stacks is NetCDF4 (on top of) HDF5, MPI-IO and POSIX. This stacking has several restrictions: Firstly, it dominantly use of POSIX as file system API. Therewith, scalability limitations implied by POSIX' strict consistency model apply. Secondly, the semantical information available in high-level I/O libraries is lost and not exploitable by the storage system. Thirdly, the loose coupling of the middleware layers with the file system lead to a replication of features within the layers; this not only increases implementation effort but the interplay of deployed optimizations may even lead to suboptimal performance.

Workflow and process automation Scientific workflows often involve a pre-processing and post-processing of data. Existing file systems, however, do not provide efficient support to notify availability of data records. In the workflow management, people often use (empty) ready files with a well-known name that are created once the data is available. Another program periodically checks for the ready file and starts post-processing once it is ready. However, due to a lack of metadata consistency guarantees it may happen that even though the ready file is available, the data is not yet. Also, many scientific domains offer data products of their simulations to their community. Therefore, they often import created data and add indexes to speed up data exploration. Efficient data management is one of the most important aspects of storage and yet standardized approaches to do it has been lacking. There have been some attempts to provide data management frameworks in the past within the storage community, but with very limited adoption [5].

Technical hints inhibit performance portability Parallel file systems often provide mechanisms that allow programmers to disclose their I/O pattern knowledge to the lower layers of the I/O stack through a hints API. This information can be used by the file system to boost the application performance, for example, through data prefetching. Unfortunately, programmers rarely make use of these features, as every file system needs other parameters to exploit performance potentially even using different interfaces. Additionally, scientific applications frequently perform small non-contiguous accesses to files using the POSIX I/O interface. This makes it impossible for them to take advantage of automatic optimizations, such as collective I/O or data-sieving enabled by the MPI I/O middleware. As a result programmers are missing the opportunity to exploit the full potential of the storage system and applications perform poorly.

Checkpointing Checkpointing is a defensive approach to recover from node failures but in the best (failure-free) case, the written checkpoints are never accessed. Historically, the required storage throughput for Exascale systems is estimated based on the system memory capacity and expected failure rate. Thus, the storage backend has to be dimensioned based on the peak checkpointing demand, a rule of thumb is that one checkpoint per hour should not take more than 10% of the possible runtime, leading to storage systems, which are able to store the complete system memory in less than 6 minutes. The conventional approach to store checkpoints therefore assumes that the storage backend bandwidth grows with the system memory size, so that storage becomes one of the most expensive components of future high-performance clusters.

Projects like the DOE funded FastForward Storage and IO project want to tackle this challenge by imple-

menting burst-buffers, which are based on expensive high-performance SSD storage and which are able to store the last recent checkpoints on SSDs, while slowly synchronizing some (not all) checkpoints with disk-based long-term storage [3].

Changes in the design of high-performance storage clusters introduced by demands of data-intensive applications offer a second, more cost-effective approach to store checkpoints. Nodes of HPC systems are more and more equipped with node-local storage to accommodate the demands of big data challenges, which could also be used to store checkpoints. Interestingly, SSDs within nodes are significantly less expensive than SSDs within backend storage environments, making them an interesting alternative to build a distributed storage environment. Further, there will be a plethora of new NVRAM technologies on the horizon that will need to be reasonably incorporated in this I/O hierarchy to balance performance and cost constraints.

Performance analysis Due to the complex interplay of hardware and software layers in the current systems, it is non-trivial to estimate performance of new accesses patterns. If the observable performance for an I/O-intense application behaves badly, it is very difficult to identify the bottleneck. While there are many tools that reveal observed hardware performance, it is barely possible to investigate the data flow in the I/O path in detail as existing systems are not including explicit support for this use case. Also, capturing and analyzing all file systems events is overwhelming as parallel files systems easily generate millions of events per day. Only the combined analysis of application and storage behavior enables us to understand I/O performance issues. It is important to realize that the impossibility to analyze this data have been responsible for the majority of I/O problems on ground-breaking installations, introducing typically weeks to months between the observation and resolution of I/O problems. The trace analytics problem of a large scale I/O installation is a textbook example of what is called big data analytics, and requires best of breed tools, for example to distribute the data reduction as in Hadoop analyses in order to avoid approaches that have failed to scale.

Performance prediction The steady growth of file system scalability often leads to novel problems that must be overcome. Being on the forefront of large-scale storage system, the identification of scalability inhibitors before a new system is built is barely possible with existing storage systems. Also, strategies to overcome these scalability hurdles can only be evaluated once a system becomes available. While predicting I/O behavior for millions of nodes is a difficult task, there is a need for an Exascale I/O simulation and modeling environment which can serve the needs of research in I/O infrastructures. A flexible environment embedded in the storage middleware reduces the costs and constraints of having real deployments with which to experiment and demonstrate improvements.

E10 – Architecture

Parallel file systems have evolved since the introduction of GPFS and Lustre; now they offer advanced management features, which are comparable to enterprise file system environments. As it typically takes around 10 years for a file system to mature and to provide the required feature set, it is not feasible to just replace these file systems by a new approach. Existing file systems include a number of design decisions which have been made for Terascale HPC environments. Unfortunately, they have never been designed to scale to hundreds of millions of file creates per second or a sustained IO bandwidth of hundreds of TByte/s.

E10 therefore takes a two-step approach and provides in the first step a number of tools and interfaces which help existing as well as new storage environments to become more efficient (see Figure 1). In parallel to this ecosystem, the E10 working group will build new core storage systems, which meet the functional requirements of future systems.

E10 Middleware

The E10 middleware described in this section is decomposed into a number of loosely coupled components, which should help existing and future file systems to become more scalable and more manageable. Most of the components' design is based on insights, which have been made individually by different researchers and for different file system implementations. Unfortunately, these ideas have not gained wide-spread adoption for several reasons. One reason is that there are now a handful of competing approaches each trying to fix suboptimal file systems but providing interfaces at different granularities. Thus applications must be adapted specifically for each prototype.

E10 is providing middleware components, which are directly adapted to the most used file systems, so that it becomes sufficient to adapt programs once to a specific feature. Furthermore, we are working closely together with the file system developers, so that

required changes can become part of the mainline source tree.

E10 Guided Interfaces: E10 guided interfaces will be implemented as a middleware layer that will allow applications to indicate expected access patterns and future use of data and to submit this information to arbitrary storage backends. This will allow the storage system to assist with performance aspects such as data distribution or re-use. The middleware will, in contrast, e.g., to POSIX hints, enable application developers to support extensibility, asynchronous behavior, directives and support for distributed environments.

Furthermore, hints can also be given after the application has been developed, so that users are able to leverage their application knowledge.

Distributed Checkpointing E10 will develop a framework that is able to distribute checkpoints within the HPC system itself and will build on the concepts of the scalable checkpoint restart library [4]. The basic idea of storing checkpoints on the nodes themselves is that the bandwidth easily scales with the problem size. A major difference to previous approaches is that it is not based on an interposing library, which can only work for a fixed setting of applications types, e.g., MPI applications. Instead, the local resource manager will create at applications start-up time a distributed, virtualized storage environment, which can be accessed in the same way as a standard file system. In contrast to a parallel file system, the environment will not support concurrent accesses to a single volume, as the different nodes can be easily coordinated by the middleware to write the checkpoint without the requirement of complicated locking protocols. The virtualization approach will also enable us to asynchronously hide the data transfer between the nodes by first buffering the checkpoints within the node.

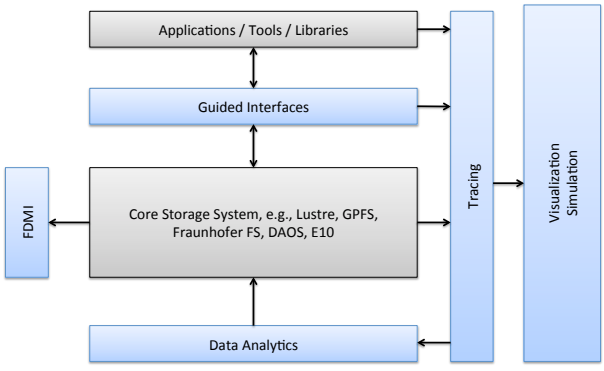

Figure 1: *Interfacing the E10 middleware. Blue components will be developed within E10.*

FDMI Plug-ins: The FDMI (File Data Management Interface) framework within E10 aims to remove parts of the complexity of previous approaches by focusing on information, which could be provided by the distributed file system logs. The standardized interfaces enable data management features to be added as flexible "plug-ins" very easily and E10 will create "plug ins" for Information Lifecyle Management (ILM), data repair, high availability and file system checking. Most third-party features within E10 in the future are expected to be developed as FDMI plugins.

Performance Analysis and Visualization: E10 will leverage existing general purpose tracing tools like Scalasca, Vampir as well as our experience in projects like the SIOX project to create an infrastructure that collects I/O events from all I/O layers, visualizes the information and analyzes the I/O events for patterns that are performance critical to be able to detect dependencies between the different storage system and application components [6][7]. The trace environment and the core storage system will provide interfaces to a data analytics framework, which helps to understand optimization opportunities and which will also provide on-line interfaces to steer applications and system configurations.

Simulation: Simulation of the I/O infrastructure will be performed under various application workloads based on "operation logs" collected by E10 partners. These real world operation logs can be edited to emulate "what if" scenarios and hence the Simulation and Modeling component is fed with various alternate scenarios within the workload. The simulation and modeling components are built from characterizations of real storage hardware and software components. Mathematical characterizations are sought for storage hardware and behavioral characterization is sought for software components. These form the predictive analysis components of I/O behavior.

E10 Core Storage System

HPC parallel file systems like Lustre or GPFS have been designed with Terascale, not Exascale computing in mind. HPC data centers therefore started to provide different solutions, like parallel file systems and key value object stores, within a single environment to support different user demands concerning interfaces and metadata performance. E10 therefore investigates architecture concepts for the core storage systems, which should be able to replace today's storage approaches in the next decade.

The basic insight has been an analysis of paral-

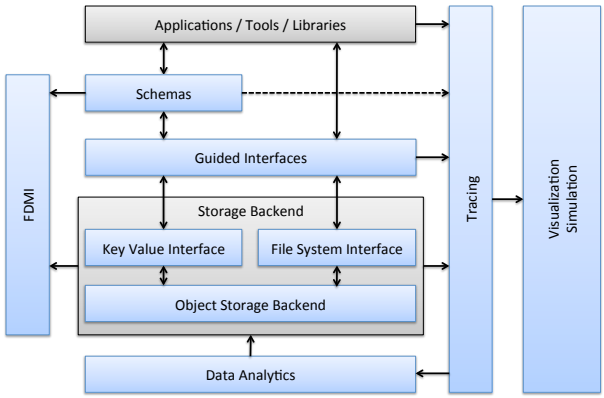

Figure 2: *E10 Core Storage System.*

lel file systems and big data environments, which revealed that the different usage scenarios are not primarily based on fundamental design differences, but on subtly different semantical constraints. The metadata performance of today's HPC file systems, e.g., is not constrained by data structures like directories, but mostly by the underlying POSIX semantics, which has been introduced to support desktop and server systems, not HPC environments. On the other hand, big data environments have been designed to overcome the metadata scalability restrictions of standard file systems, while not including the lessons-learned from decades of HPC file system development.

The E10 core storage system architecture (see Fig. 2) is based on an object-storage substrate, which builds the foundation for a key-value and a file system interface. The key-value interface is based on the standard put and get semantics of today's key value stores, so that NoSQL databases can directly reside on top of this interface without an additional translation. We intend to standardize an advanced object interface based on this concept that further provides horizontally scalable and non-posix semantics for distributed objects. The key-value function of the interface would provide the required distributed metadata semantics for objects. The file system interface will be close to the POSIX semantics, but will be optimized for high metadata create- and update rates, while defensive operations, like 'ls *' will slightly change their runtime behavior. The semantics of the key-value and the file system interface will be close enough that data stores can be concurrently accessed from both interfaces.

Complex data structures, like NetCDF- or HDF5-files, will be supported through schemas. Semantic information, which is today lost between the execution library and the file system, will therefore be available for the core storage system, so that additional optimizations become possible.

An overarching concept to support this environment at scale is the usage of a highly available RAS component, which contains configuration, data collection and reduction based on distributed monitoring concepts.

Development and Roadmap

The E10 architecture is developed and implemented as a community effort. The initial architecture has been derived after a number of requirement gathering workshops with computational scientists from North-America, Europe, and Asia until 2012 and has been in a convergence process between end of 2012 until mid of 2014. The component interfaces have been designed based on regular architecture workshops and have been opened to the community via the http://www.eiow.org web site.

HERE WE HAVE TO ADD THE ROADMAP

Conclusion

This white paper introduced the vision of the E10 workgroup on how to change the architecture and use of HPC storage environments. Changes concerning the E10 ecosystem have been designed to be independent from the core storage subsystem, so that benefits can be leveraged by existing file systems, e.g. Lustre, GPFS, or Fraunhofer FS. Significant parts of the development of the core storage environment will require time to be implemented and to become mature, so that their new concepts can help scientists to benefit from the underlying scalability and reliability enhancements.

E10 has been thus far been supported in a small way by a number of European and national projects, including the EU FP 7 MCITN 'SCALUS', the EU Deeper-project, the German BMBF SIOX project and the German DFG Transregio on Soft Matter Systems. While these projects, including additional fundings from the partners, have been sufficient to derive the basic architecture, there is huge scope for new ideas and methods to be explored and fully evaluated. E10 expects to propose a future project focusing the research into this area and targeting the standardization of API's across the international communities as part of the EU H2020 work programs for which we would very much like to have your support.

References

[1] J. Li, W. keng Liao, A. N. Choudhary, R. B. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netcdf: A high-performance scientific i/o interface," in *Proceedings of the ACM/IEEE SC 2003 Conference on High Performance Networking and Computing (SC)*, Phoenix, AZ, USA, Nov. 2003.

[2] R. Thakur, W. Gropp, and E. L. Lusk, "On implementing mpi-io portably and with high performance," in *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems (IOPADS)*, Atlanta, GA, USA, May 1999, pp. 23–32.

[3] J. Bent, G. A. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate, "Plfs: a checkpoint filesystem for parallel applications," in *Proceedings of the ACM/IEEE Conference on High Performance Computing (SC)*, 2009.

[4] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC)*, 2010.

[5] The Open Group, *Systems Management: Data Storage Management (XDSM) API (CAE Specification)*, 1997.

[6] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel, "The vampir performance analysis tool-set," in *Proceedings of the 2nd International Workshop on Parallel Tools for High Performance Computing*, Stuttgart, Germany, Jul. 2008, pp. 139–155.

[7] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler *et al.*, "Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir," in *Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing*, Dresden, Germany, Sep. 2011, pp. 79–91.