nextgenio newsletter

Welcome to the third issue of the NEXTGenIO newsletter!

NEXTGenIO at ISC-HPC 2017

PARTNER BOOTHS

F-930

J-622

A-1412

B-1332

B-1223

G-820

EVENTS

We will be present at and hosting a number of events throughout the ISC-HPC conference, as well as at the partner booths. To find out more about the project, you can find us at:

Tutorial 01: Understanding & Improving I/O Performance on HPC Systems

We are hosting this tutorial, which takes place on Sunday June 18th, 9:00am to 6:00pm in room Analog 1.

Presenters:
Keeran Brabazon (ARM), Holger Brunst (TUD), Adrian Jackson (EPCC), Tomislav Šubić (Arctur)

BoF 09: Towards Addressing Exascale I/O Requirements & Challenges

We are also hosting this BoF session, which will take place in the Kontrast room from 11:30am - 12:30pm on Tuesday 20th June.

Organised by:
Hans-Christian Hoppe (Intel), Michèle Weiland (EPCC)

Workshop on Performance & Scalability of Storage Systems (WOPSSS)

Michèle Weiland is an invited speaker at this workshop, which takes place on Thursday 22nd June at 2pm in the Gold 1 room. Her talk is titled *'Non-volatile memory for next generation I/O.'*

A Scalable Object Store for Meteorological and Climate Data

(Simon Smart, Tiago Quintino, Baudouin Raoult, ECMWF)

Numerical Weather Prediction (NWP) and Climate simulations sit in the intersection between classically understood High Performance Computing (HPC) and the Big Data / High Performance Data Analytics (HPDA) communities. Forecast operations and research at ECMWF generate approximately 100 TiB of data each day, and more than 100 TiB of this is archived perpetually. This data is in chunks, called fields, of between 1 and 20 MiB, encoded in GRIB format, and each field is uniquely identified by metadata made up of a set of key-value pairs.

For 25 years ECMWF has used the Fields Database (FDB) as a domain-specific, indexed object store as part of its multi-producer, multi-consumer workflow. The output from ECMWF's Integrated Forecasting System (IFS) is written into the FDB, from where it is retrieved by the various post-processing and archival tasks. At any time, the operational FDB contains between 4 and 5 PiB of data.

The amount of data is growing rapidly. In 1995, operations generated a total of 14 TiB for the whole year, whereas by the end of 2016 just the NWP model produced 20 TiB per hour. This stands in contrast to typical HPC practice, where roughly three quarters of I/O usage is defensive in nature, consisting of large blocks of checkpointing data.

To support the scientific objectives of ECMWF's 2025 strategy, both the resolution and diversity of the generated data must increase substantially. However, I/O performance growth has not kept pace with the available computational capacity. Both software changes and the use of novel technologies will be required to support the data volumes of the future. Within the NEXTGenIO project, ECMWF is preparing the FDB to make use of NVDIMMs.

A new version of the FDB (version 5) has been developed to overcome the limitations of the current operational system (FDB4). In FDB4 data is stored in carefully named directories, within which an index file is maintained as a flat list of the written fields. In parallel runs, MPI is used to synchronise write access to this file. Careful control of the concurrent processes is required, as only one process or MPI job can operate on the index file at any one time.

The FDB5 has an entirely new architecture. At the top level it is built with a strict separation between frontend and backend. The operational semantics are defined by the front end, and their implementation by the back end. The system indexes data hierarchically according to a configurable schema, such that each back end needs to implement the following concepts:

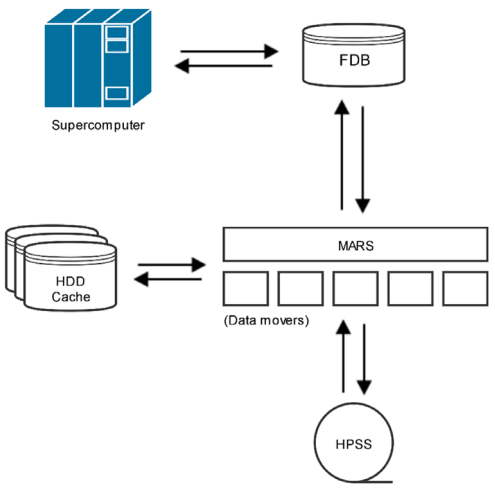
The database; this describes a larger dataset at a high level, in particular why and when the data was produced. ***The index***; a number of indexes belong to each database. These identify groups of fields that are likely to densely fill the metadata space. ***The field***; the data elements being stored. From a storage perspective these are simple byte streams.

All written data is considered to be immutable. Any writes using the same metadata result in a new entry which masks the old one, rather than overwriting it. This avoids potential data corruption associated with partially written elements and preserves the chain of events leading to the current state. Each backend is responsible for enabling simultaneous access from multiple unrelated readers and writers.

On current hardware, the FDB5 backend uses a parallel filesystem. Each database has a single table of contents (TOC). Each writing process builds its own index and data files, and once this index is flushed to disk an entry is appended to the TOC. This ensures partial or corrupt data will never become visible in case of failures.

A backend built using the Linux Non-Volatile Memory Layer (NVML) builds a branching tree in persistent memory according to the metadata key and a schema. Transactional behavior is enforced using inter-process locking. This will facilitate the integration of NVDIMMs into the workflow at ECMWF.

The newly developed FDB5 is currently undergoing acceptance testing at ECMWF. This will bring a transactional object store into the operational workflow. This object store has a strict decoupling between front- and backends which will enable the integration of novel technologies into the forecast workflow, including the use of NVDIMMs within NEXTGenIO.



A simplified overview of the data infrastructure at ECMWF

New I/O Performance Analysis for NEXTGenIO (Holger Brunst, TUD)

The NEXTGenIO project will improve I/O performance of applications by utilising NVRAM. Therefore, I/O characteristics of existing applications are recorded and extrapolated to quantify the performance gain when executed on a NEXTGenIO class system. The performance analysis tools Score-P and Vampir have been enhanced accordingly and will be explained in the following paragraphs.

Today, scientific applications use abstract I/O libraries like HDF5 or NetCDF. From a programmer's perspective, these libraries ease the handling of I/O by means of complex operations. Yet, the high degree of abstraction also complicates I/O tuning due to its deeper I/O stack. The increased complexity is addressed by a new visualisation approach with multiple layers. Now, each I/O layer of a parallel application is monitored and depicted independently. Support for the following layers is available: POSIX, MPI, NetCDF, and HDF5.

Figure 1 depicts an application using the NetCDF library for I/O. The high level NetCDF as well as the low level POSIX I/O operations are recorded and visualised. The user is able to see the interaction between different layers.

Figure 1

In this example, the NetCDF operation results in three Posix write operations. Depending on the application, the user can choose the granularity of the I/O analysis and is able to see the interaction between different layers. In addition to the I/O multi-layer support, we are developing an I/O summary which presents statistical I/O information like bandwidth or number of I/O operations. Figure 2 depicts this new view for the same NetCDF application. The I/O statistics of Figure 2 are grouped by the operation type. However, it is also possible to group by another criterion like the I/O layer.

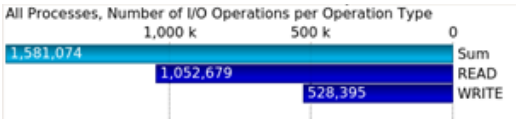


Figure 2

The I/O timeline view is another new key feature. This is a resource-oriented view and shows I/O operations for a selected time frame. The I/O timeline displays all I/O operations regarding a specific resource (file) across all processing elements as depicted in Figure 3. By means of the I/O timeline, developers are able to see concurrent accesses to a file and can identify the cause of a potential I/O bottleneck. The chart can also be switched to a mode that shows all files on the vertical axis to see when files were accessed during execution.

The new features will help to characterize the I/O behaviour of parallel applications for current systems and next generation systems deploying NVRAM. The new I/O multi-layer support can isolate program behaviour at a given level of abstraction, which is essential for the efficient usage and implementation of high level I/O APIs.

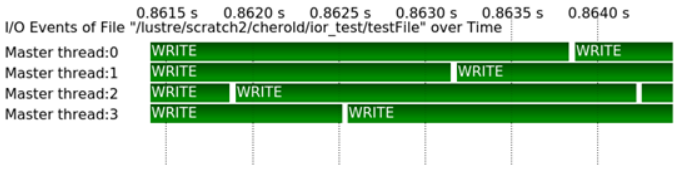
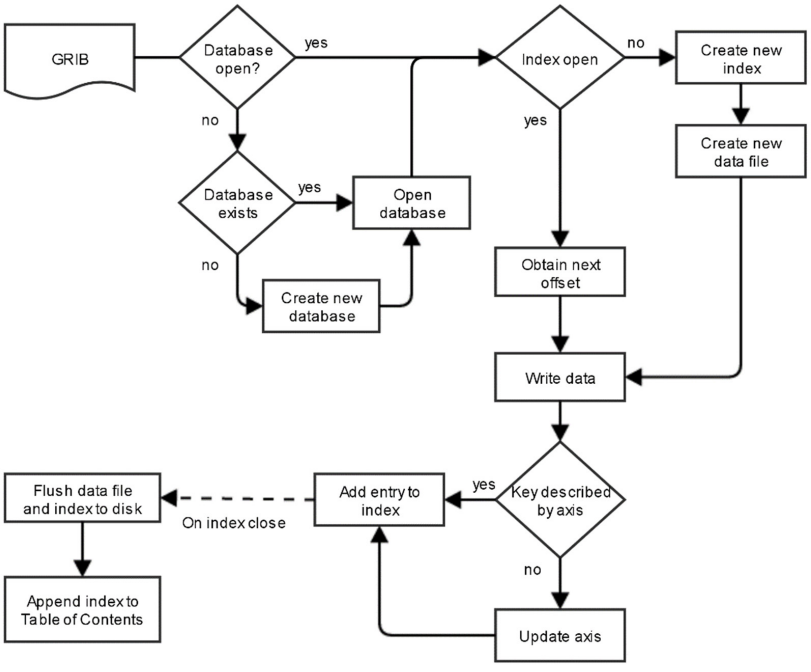


Figure 3



The write pathway in the FDB5 using the POSIX filesystem backend
(see facing page)

echofs: A Scheduler-guided Temporary Filesystem for Distributed Access to Node-local NVMs

(Alberto Miranda, BSC)

The current growth in data-intensive scientific applications poses strong demands on the HPC storage subsystem, given that data needs to be typically copied from compute nodes to I/O nodes and vice versa when calculations start and stop. In this scenario, the emerging trend of adding denser, NVM-based storage to compute nodes offers the possibility of using these resources as burst buffers to construct temporary filesystems that perform ad-hoc I/O optimisations for specific batch jobs.

In the NEXTGenIO project, we are is developing a temporary filesystem called ***echofs*** that coordinates with the job scheduler to detect and preload a job's input files into a collaborative burst buffer, which is created by virtually aggregating the NVM burst buffers available to compute nodes. The filesystem is intended to reside on compute nodes, and offer an additional storage layer between HPC applications and long-term storage that will survive for as long as a batch job needs it.

By providing this additional layer, echofs intends to fulfill several goals:

1) Hide the storage hierarchy/complexity from applications by providing a single mount point (e.g. NVM, SSDs, HDDs and long-term storage).

2) Automatically manage the migration of data between the available storage layers in response to input from the batch scheduler and applications.

3) Allow compute nodes to export local storage to be used transparently by other nodes.

4) Collaborate with the job scheduler to capture the data dependencies of the submitted jobs, and preload their data before they start.

5) Offer a POSIX file interface, with well-defined concurrent write and read operations.

Figure 1 shows an overview of the filesystem's architecture. The filesystem will distribute data across nodes with pseudo-randomized striping strategy to favor load balance, offer aggregated I/O and allow for efficient data redistributions when the number of compute nodes change between jobs. Metadata operations for persistent and not temporary files will be forwarded to long-term storage (though QoS may be applied in order to avoid congestion), while metadata for temporary files will be kept internally in compute nodes. An initial prototype for echofs will be available next year, and will allow legacy applications to transparently benefit from NVM storage.

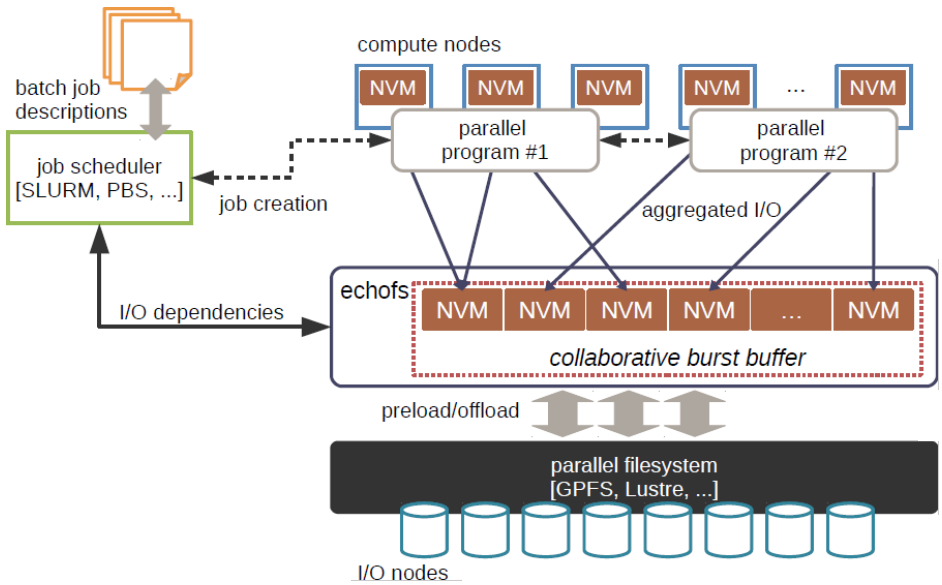


Figure 1