# R&D Studies on a Persistent Storage Buffer for the ATLAS Phase-II DAQ System

**Adam Abed Abud**[1,2], Matias Bonaventura[1,3], Edoardo Farina[1,4], Fabrice Le Goff[1]

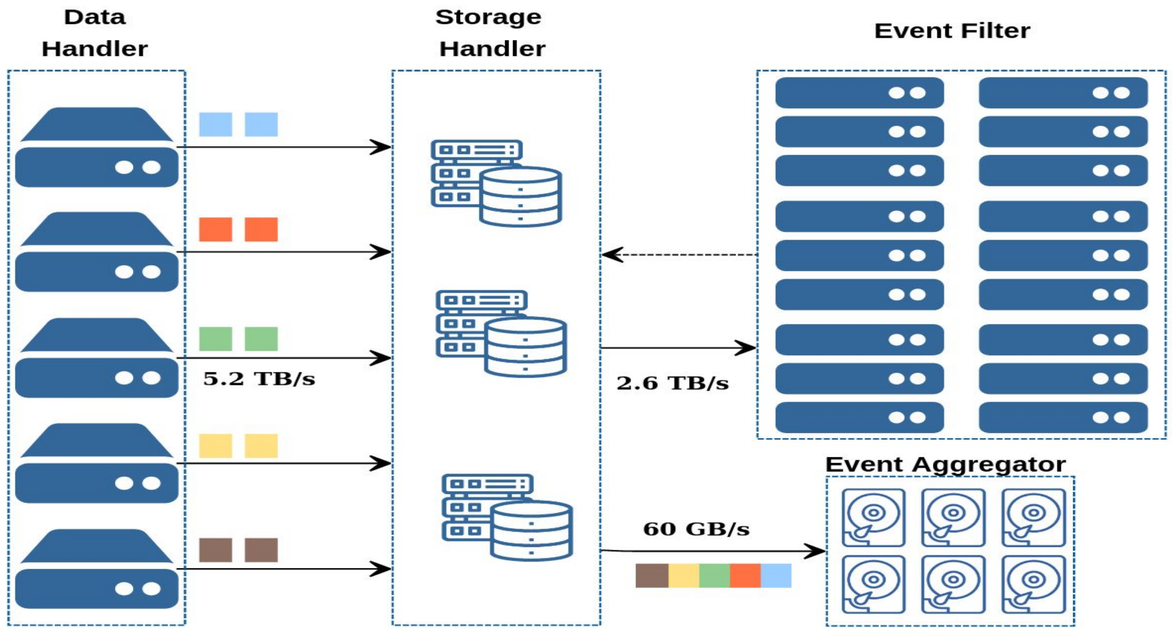
[1] CERN, Geneva, Switzerland [2] University of Liverpool, Liverpool, UK

[3]Instituto de Ciencias de la Computación UBA-CONICET, Argentina [4] Università degli Studi di Pavia and INFN, Italy

## Introduction

For the Phase-II upgrade the ATLAS experiment will upgrade both detectors and the Trigger and Data Acquisition system (TDAQ).

One of the novelty of the DAQ upgrade will consist in a large persistent storage system (Storage Handler, SH) that will have to support fragment insertion at 5.2 TB/s from 500 data producers nodes (Data Handler, DH) at 1 MHz. Filtering of the most interesting events is done by the data consumer nodes (Event Filter, EF) requesting data from the SH at 2.6 TB/s and accepted events are stored at 60 GB/s. The total capacity for the storage buffer is of the order of O(20) PB.




## Data safety

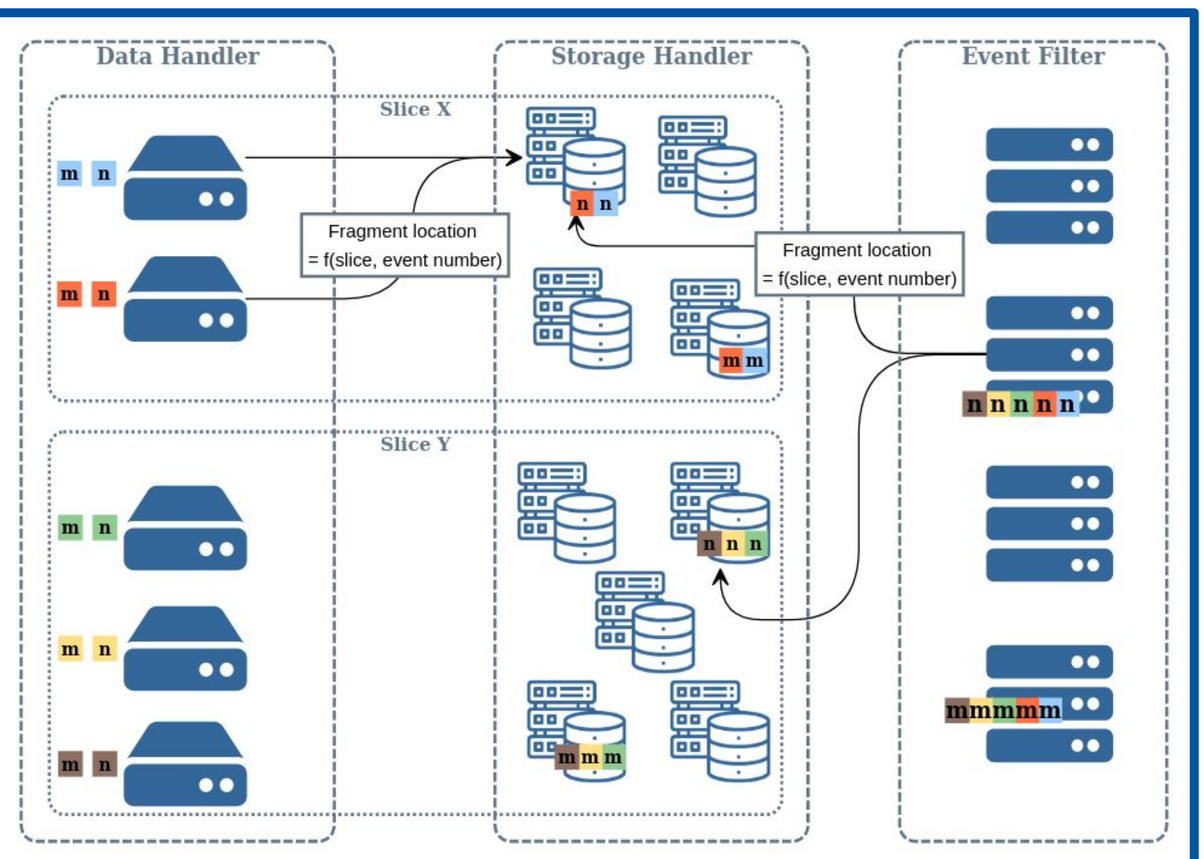
The design of the storage system must take into account possible hardware failures against which the system must protect itself. In nominal conditions, commercial NAND-based SSDs (endurance of approximately 3 DWPD) would encounter a failure after approximately 120 days of constant writing, assuming 2000 drives with 10 TB capacity and an input throughput of 5.2 TB/s. Therefore, **safety mechanisms** have to be implemented from the start. Possible solutions include:

- Higher **endurance** SSDs (e.g. 3DXPoint devices): ability to sustain higher amounts of data before a failure occurs
- **Data redundancy**: local (storage failure) and global (server failure)

Within each server, drive failures are handled with erasure coding algorithms (Reed Solomon). The system as a whole is protected against server failures by using a placement algorithm for the re-routing of the requests.

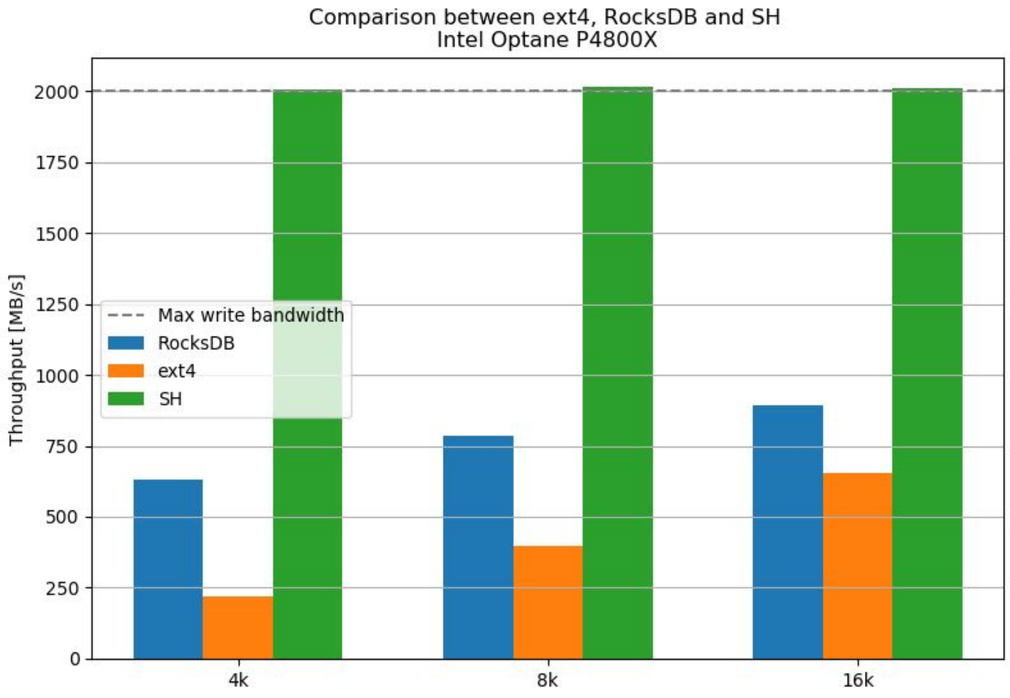
## Global placement algorithm

A key challenge of the SH is to index roughly 500 fragments at a rate of 1 MHz and distribute them among a few hundred nodes which are divided into several slices (static grouping of DH and SH nodes). To tackle this problem a shared placement algorithm was designed taking advantage of the static nature of the DAQ architecture: the same fragment always comes from the same DH node. By combining modulo operations and static rules, each detector fragment is placed in one specific server. Both DH and EF share the same placement algorithm to determine the location for each fragment. In this way, no central resource is involved and minimal communication between nodes is needed. In case of temporary server failures, another server is selected (within the same slice) based on the event ID. Later on, when the original server is back online it sends an explicit message indicating that the data is available again.




## Local data indexing

At the server level, fragments must be allocated, retrieved, deleted, and available storage space from multiple drives must be managed. These tasks are usually performed by a file system but the overheads and low performance makes this solution unusable. To solve this challenge a custom object store is being developed (SH object store) to manage metadata indexing and space allocation based on in-memory data structures, block device access, and use of kernel asynchronous I/O. The write throughput was evaluated for a file system (ext4), a COTS-based object store (RocksDB), and the custom-made Storage Handler object store (SH) for different fragment sizes. The testing consisted in measuring the write throughput when inserting blocksize of data into RocksDB or when writing data to a file on an ext4 system. The SH object store outperforms the other solutions by efficiently limiting the amount of overheads for the specific application. The resulting throughput is thus closer to the nominal bandwidth of the drive. All tests were done on the same SSD drive.




## Conclusion and Outlook

Encouraging performance results of the custom-made storage buffer for the ATLAS Dataflow system.
Next steps will focus on performance optimizations and feature development to have a more accurate prototype.

adam.abed.abud@cern.ch