

# The ALICE DAQ Online Transient Data Storage system

**F. Carena, W. Carena, S. Chapeland, R. Divià, U. Fuchs, I. Makhlyueva,  
K. Schossmair, C. Soós, P. Vande Vyvre**

CERN, Geneva, Switzerland

Ulrich.Fuchs@cern.ch

**Abstract.** ALICE is a dedicated heavy-ion detector to exploit the physics potential of nucleus-nucleus (lead-lead) interactions at LHC energies. Running in heavy-ion mode the data rate from event building to permanent storage is expected to be around 1.25 GB/s. To continue data recording even in the event of hardware failure or connection problems, a large disk pool has been installed at the experiment's site as buffering layer between the DAQ and the remote (~5km) tape facility in the CERN Computing Centre. This Transient Data Storage (TDS) disk pool has to provide the bandwidth to be able to simultaneously absorb data from the event building machines and to move data to the tape facility. The aggregated bandwidth of the TDS is expected to exceed 3 GB/s in mixed I/O traffic. Extensive tests have been carried out on various hardware and software solutions with the goal to build a common file space shared by ~60 clients, whilst still providing maximum bandwidth per client (~400MB/s, 4Gbps Fibre Channel), fail-over safety and redundancy. This paper will present the chosen hardware and software solution, the configuration of the TDS pool and the various modes of operation in the ALICE DAQ framework. It will also present the results of the performance tests carried out during the last ALICE Data Challenge.

## 1. Introduction

The ALICE experiment [1],[2] consists of 18 subsystems aiming at studying the interaction of particles colliding at the LHC. It primarily targets heavy-ion collisions (in particular Pb-Pb), but it will also be able to cope with proton-proton (pp) and proton-ion (pA) collisions. The aim is to study the physics of strongly interacting matter at extreme energy densities, where the formation of a new phase of matter, the quark-gluon plasma, is expected.

The experiment will record Pb-Pb collisions during a few weeks per year, characterized by big events (86.5 MB), large bandwidth to mass storage (1.25 GB/s), low interaction rate (10 KHz) and complex triggers. On the other hand, pp collisions will be recorded for several months per year and produce relatively small events (2.5 MB) at a high interaction rate (200 KHz), needing less bandwidth and simpler triggers with increased selectivity.

ALICE will collect physics data in a large set of detector configurations: detectors will be able to work altogether or separately, in stand-alone operation or synchronized data taking.

The ALICE DAQ system must cope with these heterogeneous requirements to transform the 1.25 GB/s aggregated throughput from the detectors into a set of recorded physics data files.

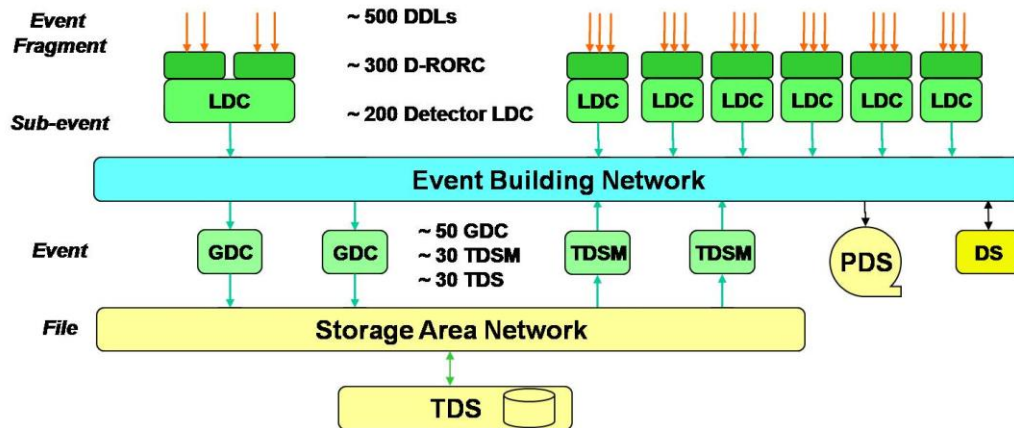


Figure 1: Scheme of the ALICE DAQ system.

The readout electronics of all the detectors is interfaced to the ALICE standard Detector Data Links (DDL). The DAQ will use more than 500 DDLs, each link being able to transport about 200 MB/s over the ~100 meters long fibres, from the detector in the experimental area to a counting room close to the surface.

Each DDL is connected to PCI-X boards, called DAQ Read-Out Receiver Cards (D-RORC), hosted by PCs, the Local Data Concentrators (LDCs). Each D-RORC hosts two DDL interfaces and each LDC can handle one or more D-RORCs.

The data read out by the LDCs are then shipped to another pool of computers, the Global Data Collectors (GDC), which build the full events. The events are then stored in files by the GDCs on a local Transient Data Storage (TDS) system, residing on a fiber channel storage area network. The files are later on migrated to Permanent Data Storage (PDS) in the remote CERN computing centre by the Transient Data Storage Manager (TDSM) machines.

The DATE software [3] is a distributed system consisting of several processes executed on every node of the DAQ system where it manages the dataflow and the control.

Some DAQ Services (DS) machines provide various central facilities to the other computers, including configuration database, publish and subscribe information servers, monitoring and logging repositories. These important nodes also run the DATE runControl, responsible for synchronizing all the DATE processes.

The focus of the present paper is the design and performance of the Transient Data Storage (TDS) system.

## 2. A Global Transient Data Storage (TDS) system

### 2.1. Data Buffering and Recording

All data recorded by the GDCs have to be written to Permanent Data Storage (PDS), which is the tape facility in CERN's computing center, more than 5 km away from the experiment site. To guarantee smooth data taking, the data acquisition part has to be decoupled from the actual data transfer to protect it from data back-pressure caused by latencies or problems with the connection or in the tape facility. Therefore a Transient Data Storage (TDS) of 50 TB buffers the files written by the GDCs and provides 12 hrs of autonomy in case of a complete outage of data transfer to tape.

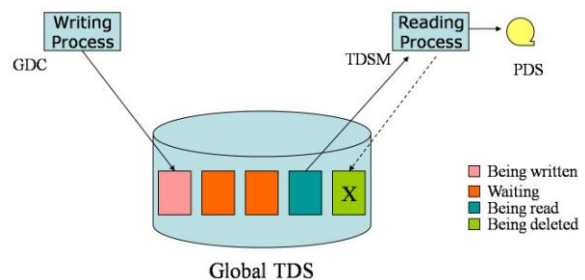


Figure 2: Scheme of the global TDS.

From the storage point of view the DAQ system consists of data storage “providers” and two sets of clients, writers (GDCs) and readers (TDSMs).

The writing process creates new files, fills them up to a certain size and closes them. The reading process opens files, transfers the content to the tape facility and deletes the file after the successful transfer (see Figure 2). The TDS has to be implemented as a shared file space, because the writing and reading processes are not on the same machine.

## 2.2. File system selection

We considered several options for the storage system, such as Network Attached Storage NAS (e.g. NFS, AFS), Storage Area Network SAN (attached SCSI) and Cluster File System (CFS) solutions.

All machines have 1 Gbps Ethernet network links to a non-blocking switch, but this link is nearly saturated by the event building traffic. Therefore all NAS solutions were dropped for performance reasons. If NAS was used, all transfers to storage would subtract bandwidth from the communication between GDCs and LDCs and therefore have a negative impact on the event building rate. The usage of two independent network interfaces turned out to not solve this problem either, because the Linux kernel used runs the network interface as a single thread and in case of two network cards hits the CPU limit. TOE (TCP Offload Engine) cards or other network technologies could be a solution but were not considered for cost reasons.

A physical device cannot be shared between reading and writing processes, because they run on different machines, thus excluding pure SAN solutions and the idea of sharing SCSI devices.

It was therefore decided to use two distinct networks for event building and data storage and to use a Cluster File System based on a Fiber Channel SAN infrastructure. In this case the total available bandwidth is limited by the number of storage devices on the SAN and the throughput achieved depends on the performance and the allocation algorithms of the Metadata Controller responsible for block allocation and journaling all transactions. The communication with the Metadata Controller only adds little overhead to the network traffic while the data is moved over the Fiber Channel interface causing no significant CPU load. The machine can therefore perform event building at maximum possible speed without being slowed down by the storage.

The selection criteria for the storage solution were stability and flexibility, vendor independence and its performance. The file system has to support up to 80 clients, each with multiple concurrent streams but also cope with the problem of mixed access patterns. The writing clients provide an aggregate linear writing stream of 1.25 GB/sec, the reading clients cause a similar linear reading stream. On the level of the TDS these two streams are mixed and result in a data I/O of more than 2.5 GB/sec in random-access patterns, which, if passed on, causes hard disks to lose more than 90% of their throughput (see Section 4.1. ). A key factor in choosing the right storage solution was therefore its performance for mixed-i/o access.

## 2.3. Solution chosen

Hardware-wise we based the installation on disk arrays from Infortrend (type A16F-G2422) offering 16 SATAII disks, RAID6 and two Fiber Channel 4G connections in a 3U chassis. The SAN switches are non-blocking 16 port 4G switches from Qlogic (type SANbox SB5602-20A), the host adapters are 4G cards from Qlogic (type QLA2340). The SAN switches are stacked using 10G links.

StorNext (v3.01) from Quantum Inc. has been chosen as Cluster File System solution.

## 3. SAN, Affinity and CFS Setup

### 3.1. A Rack-oriented SAN

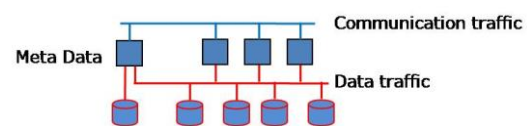


Figure 3: A SAN based Cluster File System.

Presently the Storage part of the DAQ consists of 3 Racks, each containing 5 GDCs, 4 TDSMs and 6 storage arrays, all connected to the same Fiber Channel switch. The 3 switches are stacked using 10G copper uplinks (see Figure 4).

This configuration allows full end-to-end throughput within the same rack/switch and also sufficient throughput between racks.

The setup of the CFS and the configuration of the TDSM will be made in a way to try to confine data traffic within a rack and minimize communication between them. This way there will be no “hot spots” or bottlenecks in the SAN infrastructure.

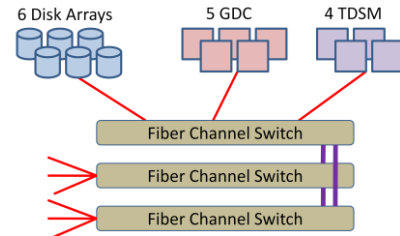


Figure 4: SAN setup with 3 Racks.

### 3.2. Affinity

As mentioned before it is essential for the performance to separate writing and reading streams to avoid artificial randomization of the traffic on disk level. Most Cluster File Systems combine all available devices in a global file system using RAID-0, which represents the worst case because any traffic is bound to use all available disks. Some Cluster File Systems use other scheduling mechanisms to direct traffic away from busy disks or similar. Still, this is not sufficient to guarantee a throughput of 3 GB/sec.

The Cluster File System “StorNext” from Quantum offers a special type of block allocation schema called “Affinity” which allows the user to combine, in the same file system, a global view across all storage (similar to RAID-0) with the ability to address individual storage elements.

Thus it is easy to assign individual storage elements (represented as directories in the global file tree) to certain data streams. This way writing streams and reading streams are physically separated, thus running a maximum throughput with no randomization, whilst still having the possibility of global access from every machine (see Figure 5).

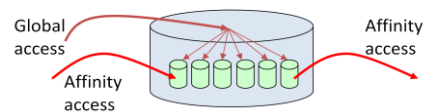


Figure 5: StorNext Affinity.

### 3.3. Construction of storage containers

There are three steps from individual disks to the final file system as it is being used right now. The first step is to combine the 16 disks ( $D_1$  to  $D_{16}$ ) in one array into four RAID-6 logical volumes ( $L_1$  to  $L_4$ ) of four disks each. Having six disk arrays per rack gives 24 logical volumes  $L_1$  to  $L_{24}$ . This is done by the array firmware.

Over three racks, the global SAN now consists of  $24 \cdot 3 = 72$  devices seen by all clients. In the StorNext CFS configuration these SCSI devices are now combined even further: three logical volumes from three different arrays are combined to form one RAID-0 Container, resulting in 8 Containers  $C_1$  to  $C_8$  per rack, thus 24 Containers in total.

Eventually all Containers are combined to form the global file system. In this global filesystem a directory is created for every container and exclusively associated with it, which allows processes to address the container directly (“Affinity Container”), see Figure 6.

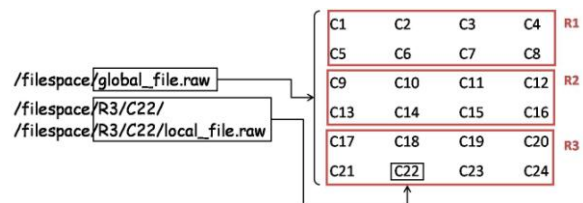


Figure 6: Affinity Containers.

In this configuration different streams into different directories actually go to different disks and the traffic is balanced on the three arrays whose volumes compose the Container.

#### 4. Performance measurements

##### 4.1. Single array volume

The performance of a single volume consisting of 4 disks in RAID5 and RAID6 was measured by using it as directly-attached FC-SCSI device formatted with a XFS file system. Several measurements were taken varying block sizes and buffer sizes for linear write, linear read, random write and random read. The following figures show typical results.

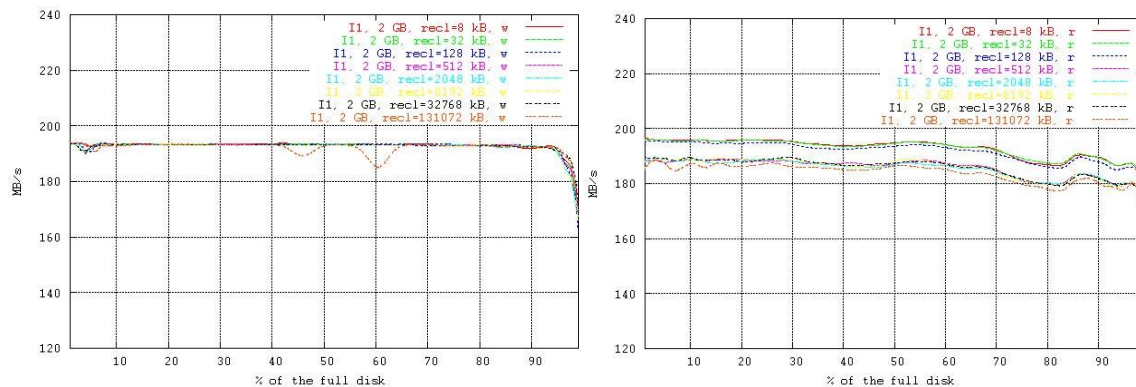


Figure 7: Linear write and read to local RAID5 volume.

These results were achieved on a 2GB Fiber Channel SAN, the maximum throughput being in the range of 190-200 MB/sec. The plots in Figure 7 show the linear writing and linear reading performance as a function of the fill level of the logical volume. It is clearly visible that the writing performance is constant for different file- and block-sizes and only starts to decrease up to 20% at a fill level of 95% (see Section 4.2. ). For reading the throughput is also nearly constant up to a fill level of 70% where variations of at most 10% start.

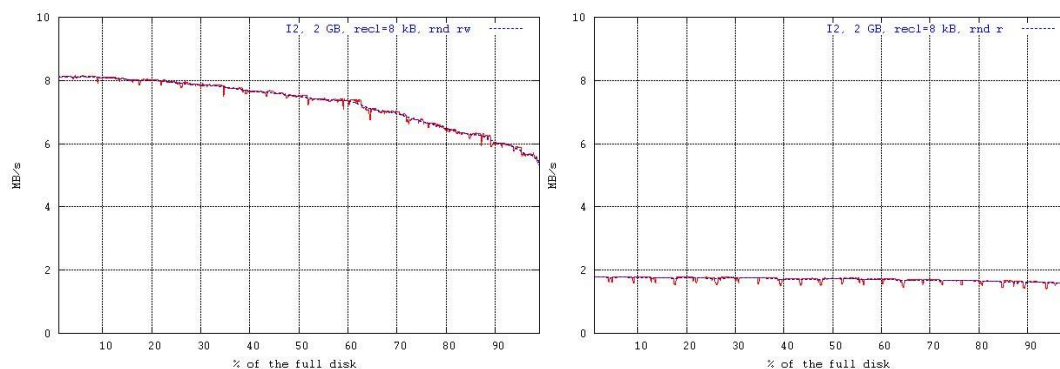


Figure 8: Random write and random read.

The plots in Figure 8 show the same measurements taken with random writing and random reading streams for the same volume. The absolute numbers (max 8 MB/sec writing and less than 2 MB/sec reading) illustrate clearly that a separation of reading and writing traffic is needed.

##### 4.2. Performance penalties: The 80% Effect and Array Rebuilds

There is a striking consistency between all measurements: the performance drop at fill levels above 80%. This can be explained by simple disk geometry: it takes longer to position the head and therefore the over-all throughput goes down.

Some volumes show this effect starting at 60% already which goes back to hardware problems (bad disks, bad array backplane or controller scheduling problems), see Figure 9.

In case of disk failure the volume continues running in degraded mode, re-calculating each block from parity. In this case a drop of 15-20% in performance is observed.

Once the faulty disk is replaced the array controller automatically launches a rebuild of the volume, restoring the parity and data blocks. The performance loss and rebuild time depend on the controller settings (rebuild with high or low priority). With high priority it takes about 45min to rebuild a 1 TB volume, however, during this period its performance is reduced by about 95%. Rebuilding with low priority keeps performance levels up at about 70% but it takes 4 to 5 hrs to complete, depending on the parallel usage.

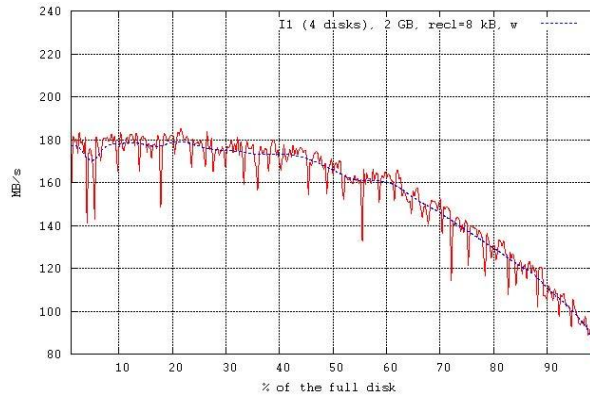


Figure 9: Writing Performance on a bad setup.

### 4.3. CFS Affinity Containers

After building the Cluster File System and combining the Logical Volumes to Affinity Containers, the measurements of linear write and linear read were repeated. Because the Affinity Containers allow separation of read and write traffic, random access patterns were not tested because it is no longer a realistic mode of operation. We concentrated on multiple streams per client simulating writing into multiple files or reading from multiple files in parallel.

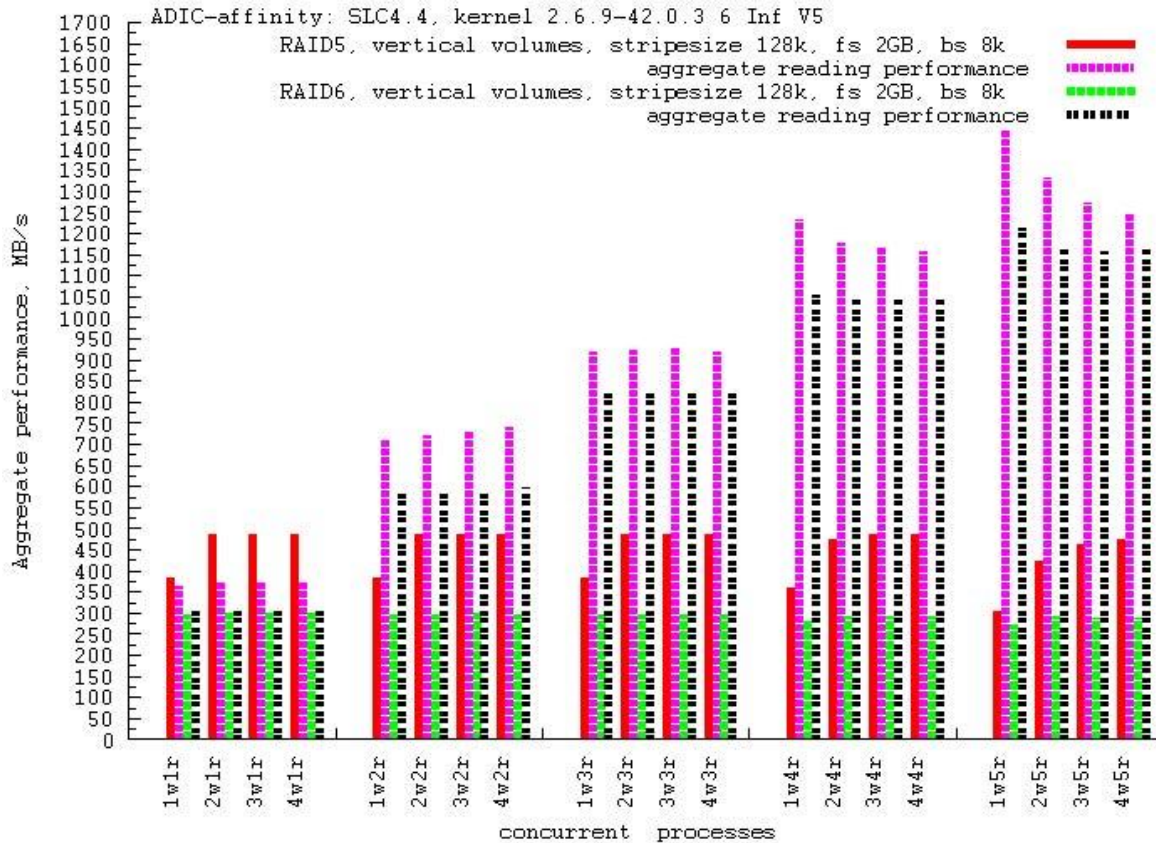


Figure 10: Performances of an Affinity Container.

In this test the typical production scenario was used. All writing streams went into the same container while each reader was attached to its own.

From the performance plot (Figure 10) one can deduce the following results: in writing mode the performance per container is limited by the Fiber Channel bandwidth for RAID5 containers. Because every container consists of 3 volumes (i.e. has three fiber connections) and is receiving traffic from multiple machines the maximum achieved throughput exceeds 400MB/sec. Writing into RAID6 containers is limited to around 300MB/sec by the array controller and parity generation. In reading mode one client also nearly saturates the fiber connection (375 MB/s), the aggregated reading performance scales linearly with the number of reading clients as they connect to different containers.

#### 4.4. ADC7 results

During the last ALICE Data Challenge (ADC7) tests the Affinity Containers were used in production-like operation. All GDCs write into one container until it reaches its high-water mark which signals the GDCs to move on to the next available one. The full container is then queued for transfer and eventually assigned to a TDSM machine which empties it moving all files to the tape facility. The empty container is then made available again to the GDCs. During the tests all three racks were used, so all 24 Containers and all 12 TDSM file movers.

Figure 11 shows the throughput on the network ports of the tape system, incoming traffic being the file moved from the CFS, outgoing traffic being the data moved to tape.

This plot is an example of the transfer rates we observed and shows that we reached peak throughputs of more than 1.5 GB/sec moving data out from the CFS, being in balance with the incoming data rate. On CFS level we therefore exceeded an aggregated throughput of 3 GB/sec.

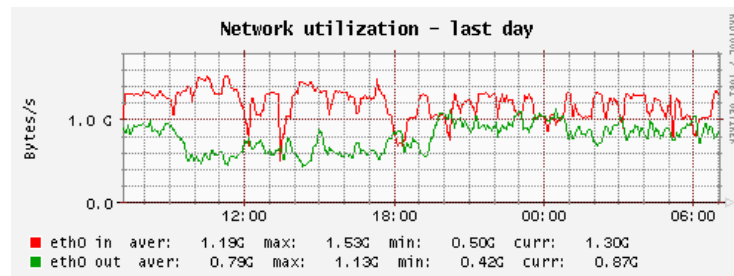


Figure 11: Performance during ADC7.

#### 5. Closing Remarks

During the ALICE Data Challenge 7 we proofed the stability and performance of the chosen CFS-SAN solution over a period of more than a month. The achieved data throughput was outstanding given the complexity of the application.

Our setup using many unique features of the Cluster File System software package and the imagination and creativity put in the choice of hardware and system setup led our nomination for the SNW Storage Award by Quantum Inc.

#### References

- [1] ALICE Collaboration, Technical Proposal, CERN-LHCC-1995-71
- [2] <http://aliceinfo.cern.ch>
- [3] <http://ph-dep-aid.web.cern.ch>