Design of a Resilient, High-Throughput, Persistent Storage System for the ATLAS Phase-II DAQ System

Adam Abed Abud, Matias Bonaventura, Edoardo Farina, and Fabrice Le Goff

May 18, 2021

Outline

1. ATLAS TDAQ overall design and dataflow requirements (Run 4)

2. Challenges for the dataflow storage system
   a. Indexing
   b. Fault tolerance
   c. Local storage management

3. Storage, network and integrated performance evaluation

4. Conclusions

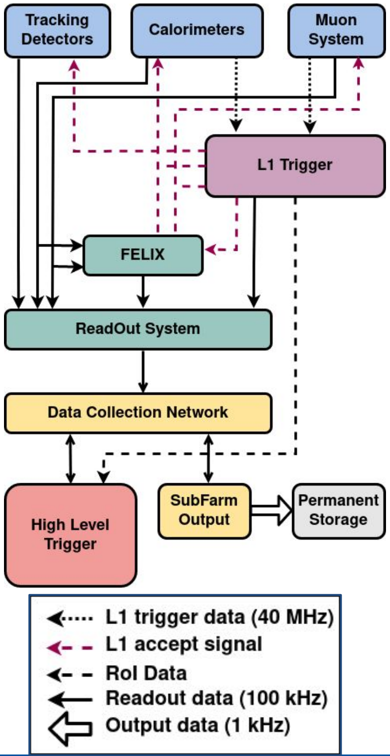
Overview of the ATLAS TDAQ Architecture Upgrade

TDAQ Phase-I (Run 3)

~40 MHz

~100 kHz

~1 kHz



HL-LHC

$\langle \mu \rangle \simeq 200$

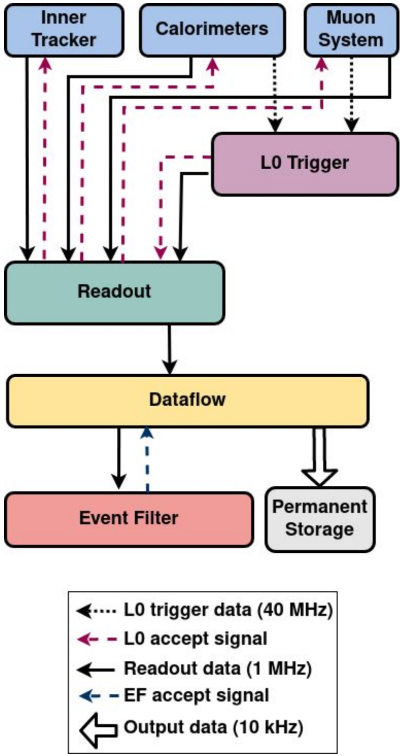
x10 higher trigger rates

TDAQ Phase-II (Run 4)

~40 MHz

~1 MHz

~10 kHz



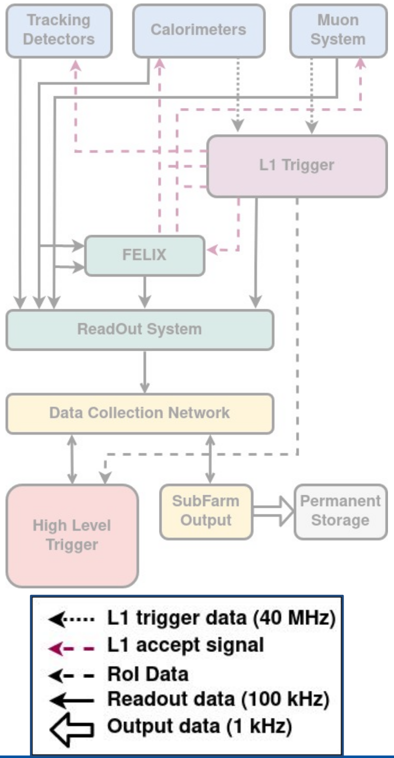
Overview of the ATLAS TDAQ Architecture Upgrade

TDAQ Phase-I (Run 3)

~40 MHz

~100 kHz

~1 kHz



HL-LHC

$$\langle \mu \rangle \simeq 200$$

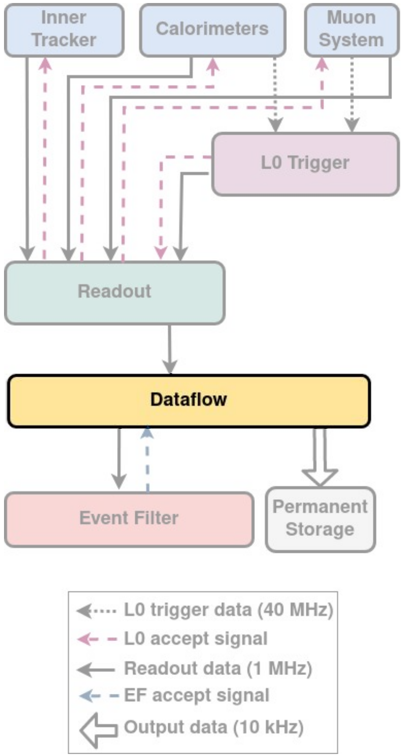
x10 higher trigger rates

TDAQ Phase-II (Run 4)

~40 MHz

~1 MHz

~10 kHz



Requirements for Phase-II TDAQ Dataflow

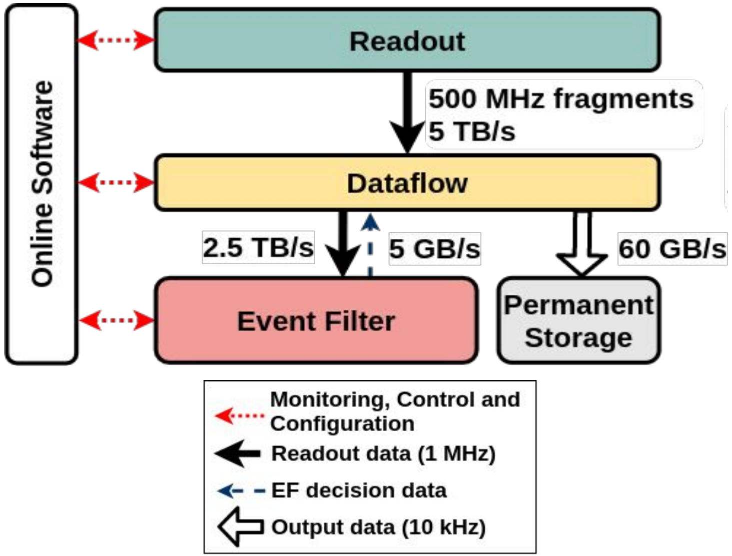
- Dataflow provides persistent buffer for readout data, before and during event filter processing, and for selected events data

- **Capacity requirements:**
  - Readout buffer: 5.2 TB/s x 10 minutes = ~3 PB
  - Selected events: 60 GB/s x 48 hours = ~10 PB

- **System size determined by the throughput requirements:**
  - Writes+Deletes: 500 MHz fragments (5.2 TB/s)
  - Reads: 230 MHz fragments (~ 2.6 TB/s)
  - Selected events: 10kHz events (60 GB/s)
  - **Total throughput of ~7.8 TB/s**

- Dataflow has a **downtime budget of 0.02 %**

DAQ components



- SSD performance projection for 2025
  - ~1800 SSDs will be needed
  - Can provide up to ~36 PB of storage

Challenges for the Dataflow Storage System

Data Indexing

Efficient indexing of fragments distributed across a few hundreds servers

Fault Tolerance

Server and drive failure safety mechanisms to achieve 99.98% service availability and data safety

Local storage management

Efficient management of local drive capacity and throughput

Challenge 1: Global and Local Fragment Indexing

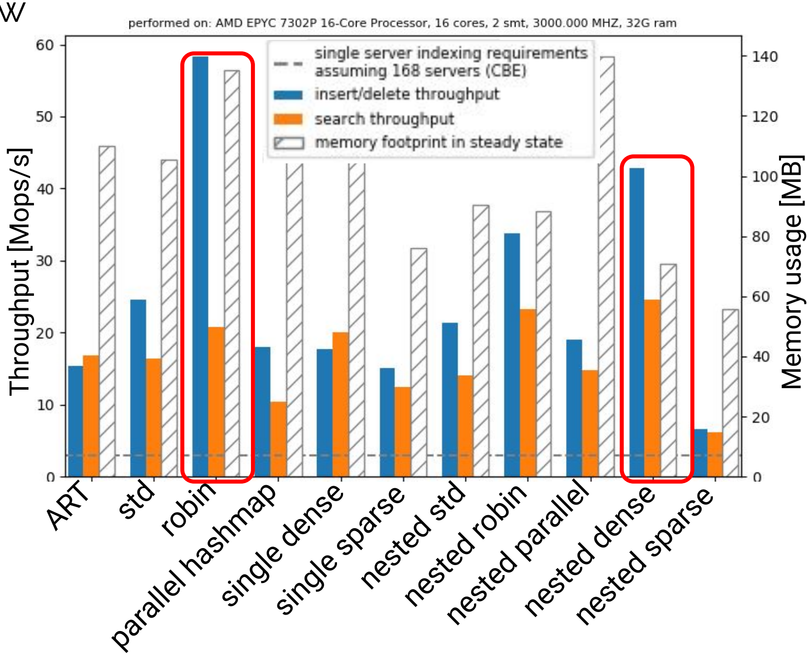
- 500MHz fragments must be software-addressable by a few hundred servers
- Parallel Distributed File Systems are not suitable for ATLAS Dataflow [1]

- **Idea:** take advantage of the static nature of ATLAS Dataflow

- **Global indexing:** static placement algorithm
  - Each fragment is associated to an ordered subset of storage servers (in case of server failures)
  - Shared by writers and readers
  - Minimum communication, no central resource

- **Local indexing:** efficient in-memory data structures
  - Key: (runId, evtId, subdetId) ; value: physical location
  - Evaluated several tree and hash structures
  - Two setups: single and nested structures

Local indexing structures performance



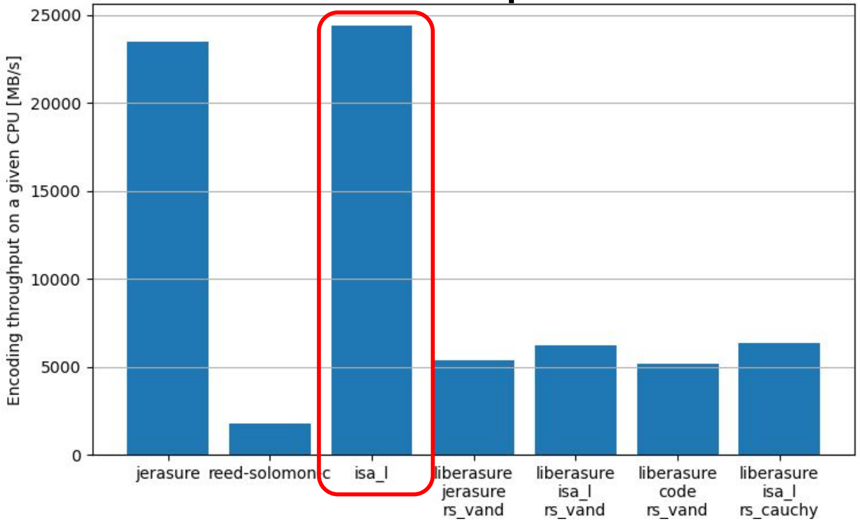
Challenge 2: Server and Storage Fault Tolerance

- NAND-based SSDs endurance: 3 DWPD -> failure after 230 days
  - Mitigation: Intel~~/Micron~~ 3D XPoint SSDs, endurance up to 100 DWPD

- **Global fault tolerance:** protect from server failures
  - Upon failure a different server is used
  - Server failure:
    - No data loss, but temporarily unavailable
    - Temporary performance degradation

- **Local fault tolerance:** protect from device failures
  - Data redundancy at the server level
  - Hardware RAID (max 3+1): 33% overhead
  - Reed Solomon (max 5+1): 20% overhead
    - Fragment splitted in 5 shards + 1 redundancy
    - Each shard at the same offset of each drive

Encoding performance of Reed Solomon implementations

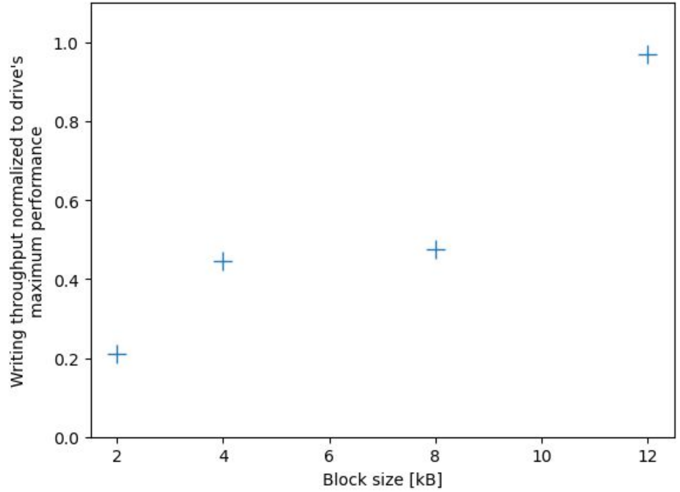


- No extra network traffic is generated to distribute redundant data across the system
- Impact on CPU and storage overhead is kept to a minimum

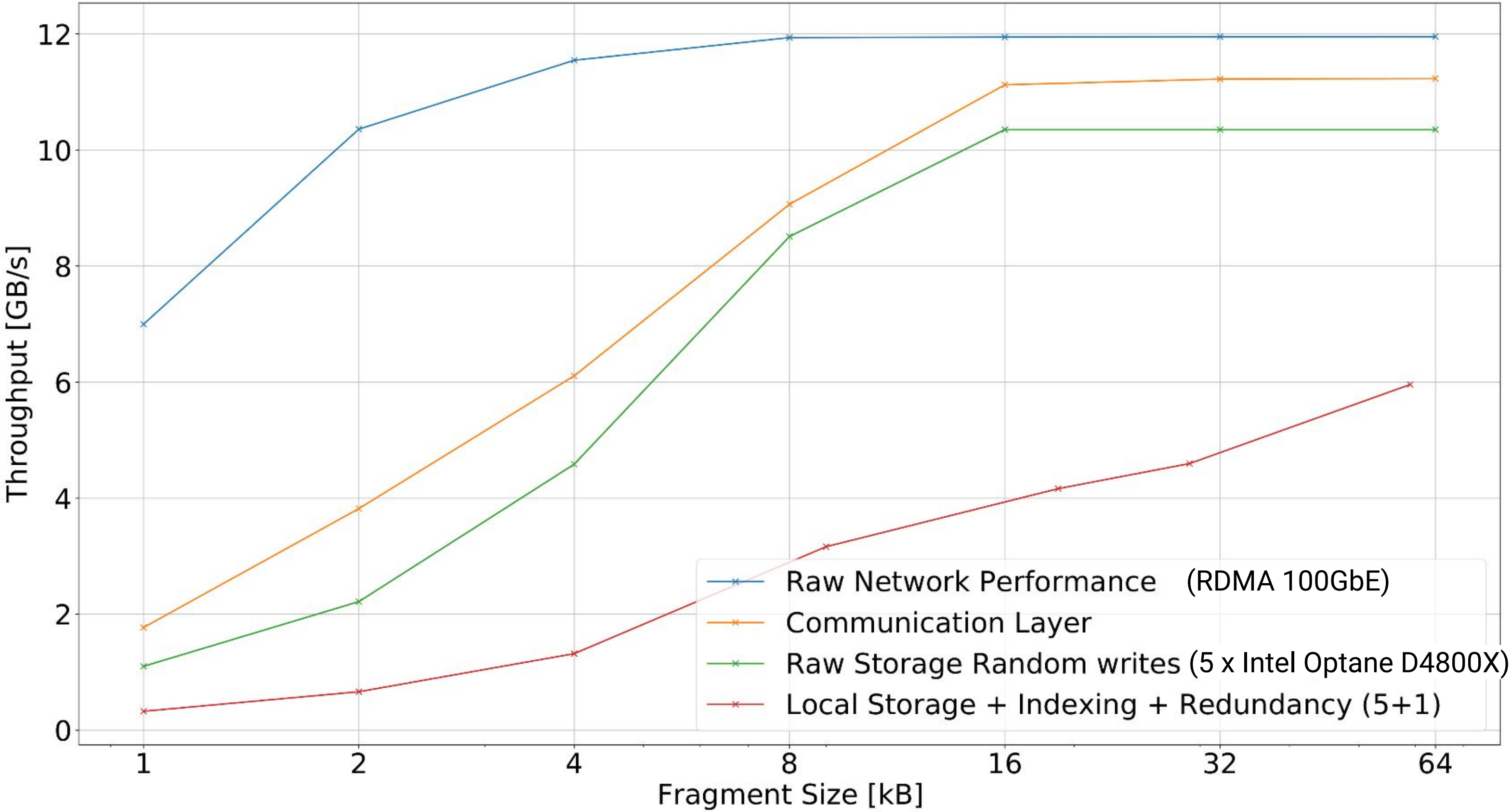
Challenge 3: Local Storage Management

- File systems (ext4, xfs, etc) not suitable for Dataflow requirements (overhead of file metadata)
- RocksDB (key-value store): 50 % of the nominal drive bandwidth was achieved (write amplification of 2)

- **Solution:** an in-house implementation is in development
  - Skip the overhead of filesystems: direct asynchronous I/O (libaio vs SPDK)
  - Our use case is simpler than generic FS (e.g. concurrency)
  - In-memory metadata management
    - Binary tree keep track of unused space
    - Log of reusable deletes blocks
  - Server failure mechanism:
    - Scan full drive to reconstruct metadata indexing
  - Writes are buffered to maximize throughput



Storage, network and integrated performance



Conclusions

- Dataflow is a high-throughput, large-capacity distributed storage system for ATLAS Phase-II upgrade

- Many challenges are being addressed:
  - High writing and reading throughput: 7.2TB/s
    - Aiming at RDMA 100Gb/s network paired with PCIe4 SSDs drives
  - Indexing data at high rates: 500MHz fragments
    - Global static placement algorithm
    - Local efficient in-memory indexing structures
  - Ensuring data safety: redundancy mechanisms
    - Reed-Solomon showed to be efficient and flexible
  - Efficient access to local storage
    - Direct block device access with asynchronous libraries
    - In-memory structures to manage space allocations

- Encouraging good performance of individual components
  - Integrated system needs further optimization
  - Next generation of PCIe4 SSDs promises to provide two to three times more throughput

Thanks from the Phase II TDAQ Dataflow team

Questions?