CERN

Automated load balancing in the ATLAS high-performance storage software

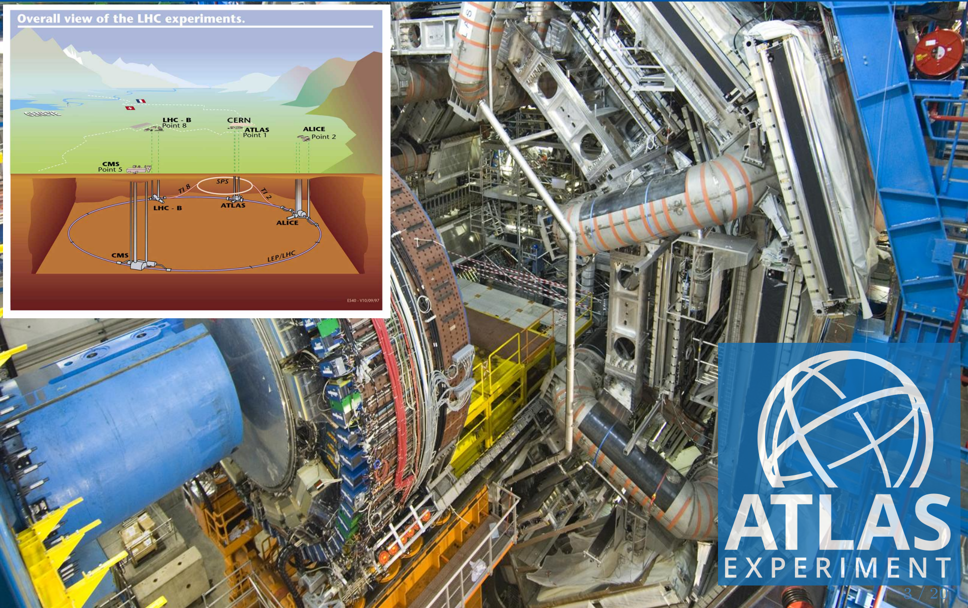
Fabrice Le Goff[1] Wainer Vandelli[1]

On behalf of the ATLAS Collaboration

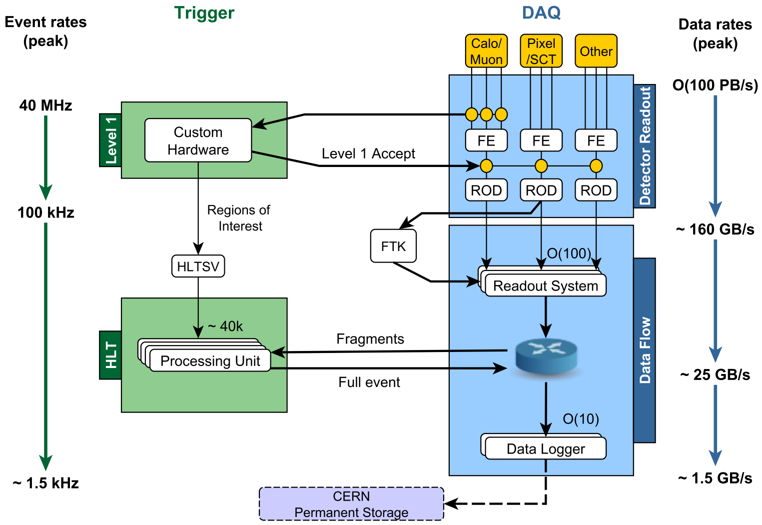
[1]CERN

May 25th, 2017

The ATLAS Experiment

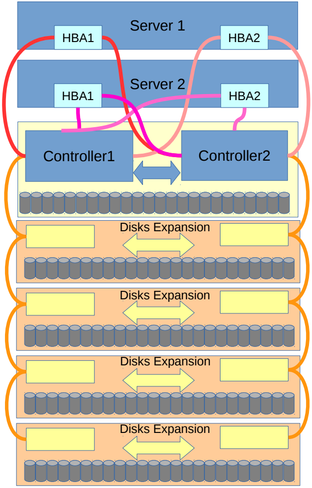


ATLAS Trigger and Data Acquisition System



Data Logger System

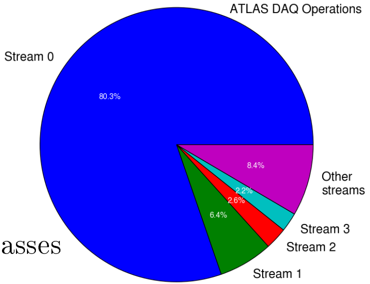
- Transient storage system to:
  - Decouple online and offline operations
  - Cope with disruption of permanent storage service or its connection
- Scale-out system, currently:
  - 4 local-attached storage, 2 servers each
  - 500 HDs, 430 TB, 8 GB/s
  - Fully-redundant: no data loss caused in 2016



HBA: Host Bus Adapter

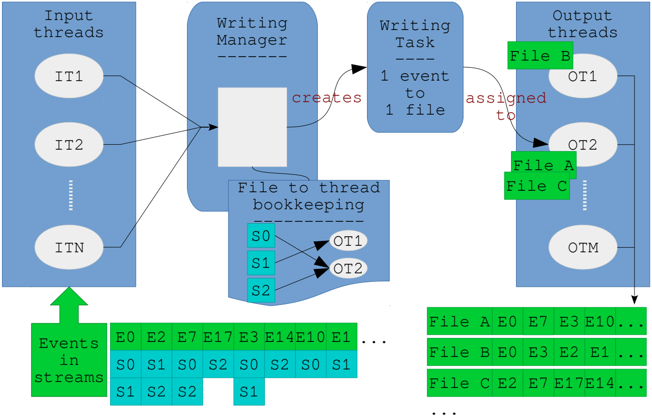
Software

- Distributed in-house application (C++)
- Tasks:
  - Receive selected events data
  - Write data to disks
  - Compute data checksum: file-by-file Adler32
- Data-driven: events are distributed in classes called streams
  - One file by stream
  - New streams appear roughly every minute
  - Stream distribution may vary rapidly
- Workload not uniform at all, cannot be fairly distributed
- Multi-threaded through task-oriented framework
- Another independent application sends the data to permanent storage



Typical Stream Bandwidth Distribution

Threading Model



- The workload distribution is controlled by the file-to-thread assignment policy
- The application performance can be limited by one thread

Current Assignment Policy

- Round-robin: each new file is assigned to the next thread in a circular thread buffer
- Simple implementation, very low overhead
- Deterministic behavior but events come with no specific order: non-deterministic assignment of files to thread
- The application's instantaneous performance is not predictable:
  - The assignment of major streams to the same thread will degrade the application performance

Problem

- Modification in the operational conditions: higher throughput, different stream distribution

	2015	**2016**
Peak throughput	1.4 GB/s	3.2 GB/s
Stream distribution	S1: 80 % S2: 6 % S3: 3 %	S1': 70 % S2': 7 % S3': 5 %

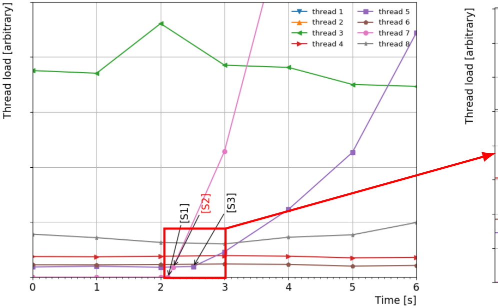
- Random assignment of major streams to the same thread will now degrade the application performance
- Synthetic test confirmed performance degradation:

Conditions	Writing rate	Performance Loss
No joint assignment	865 MB/s	reference
S1' and S2' together	797 MB/s	- 8 %
S1', S2' and S3' together	760 MB/s	- 12 %

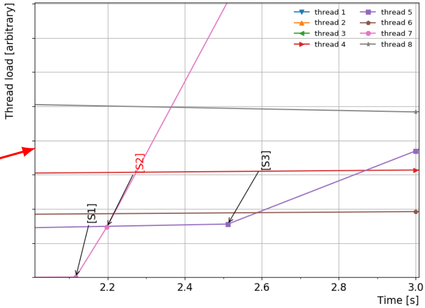
Weighted Assignment Policy

- A new workload distribution strategy was needed to restore performance and predictability
- Requirements:
  - Data-driven: e.g. cannot assume any pattern in stream distribution
  - Responsive: must cope with rapid evolution of stream distribution
  - Low CPU and memory footprint
- Idea:
  - Compute a load for thread: last-N-second sliding window of amount of processed data
  - Assign a new file to the thread with the lowest load

Weighted Assignment Policy: Step 1



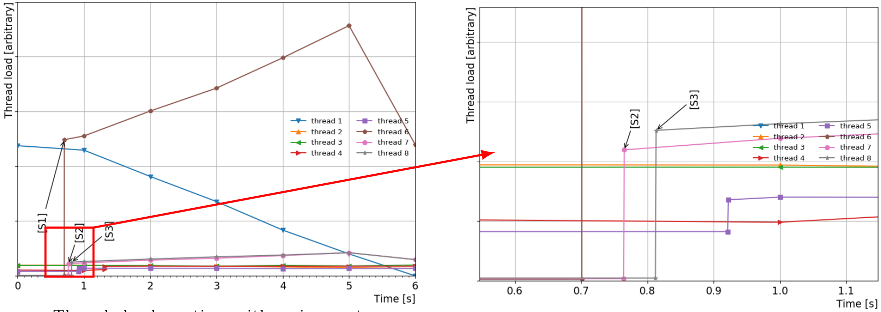
Threads load vs. time with assignments



Zoom on the assignments: problematic in red

- Real-time load is ineffective for close-enough assignments
- Reducing sliding window length: but cannot be too small, would be too sensitive to local fluctuations (typical: 5 seconds)
- Another component needed to be added:
  - Compute a load for the *streams*: same sliding-window amount of processed data by class of *streams*
  - Add the stream load to the thread load upon assignment

Weighted Assignment Policy: Step 2



Threads load vs. time with assignments

Zoom on the assignments

- Decisions are reflected immediately: the likelihood of a thread to be selected again just after decision is inverse proportional to the load of the assigned stream

Testing

- Test in controlled environment with emulated data flow:
  - Stream distribution and upstream event processing time emulated from 2016 monitoring data
  - No wrong decision for + 40-hour runs

Policy	Writing rate	Performance Gain
Round-robin	865 MB/s	reference
Weighted	882 MB/s	+ 2 %

- Test on the actual ATLAS TDAQ infrastructure
- Used during ATLAS commissioning tests and cosmic data taking sessions

Conclusion

- The transient storage system of ATLAS TDAQ is a key component enabling for decoupling of online and offline operations
- Its workload is heavily unbalanced and cannot be fairly distributed
- In 2016 a new strategy was required to handle recent changes in operation conditions
- New workload distribution strategy: sensitive and self-adaptive to fast-evolving operation conditions and modifications of the event selection process
- Validated in both test and production environments: proved to better use the parallel processing capabilities of modern CPUs for our workload
- This development will be part of the 2017 data-taking session

www.cern.ch

2015 Real-time Streams Writing Rate

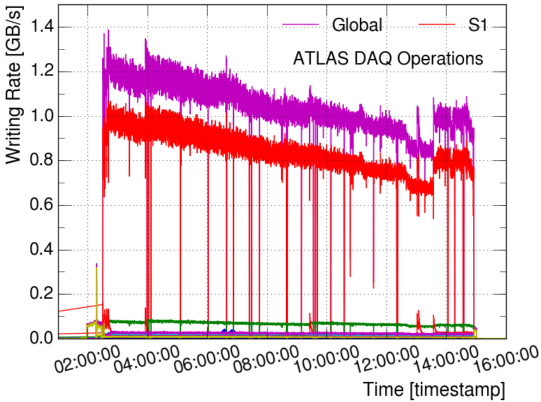


Figure: Instantaneous stream bandwidth for data collected on 28/10/2015. All streams are shown and each line represent a different stream. The highest line labeled "Global" is the sum of all streams representing the total bandwidth of selected events data.

2015 Stream Bandwidth Distribution

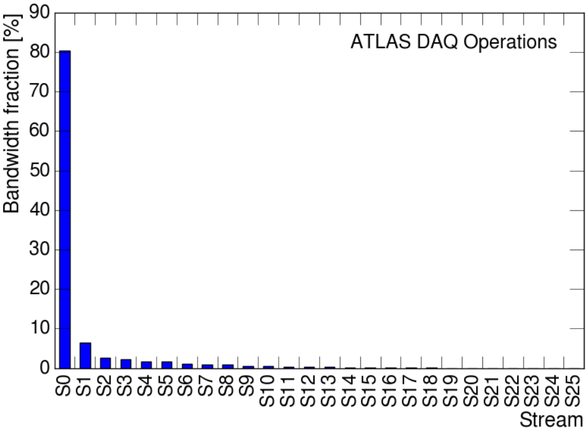


Figure: Stream bandwidth distribution for data collected on 28/10/2015. Each bar represent the fraction of the total bandwidth for one stream over the considered period.

2015 Stream Bandwidth Distribution

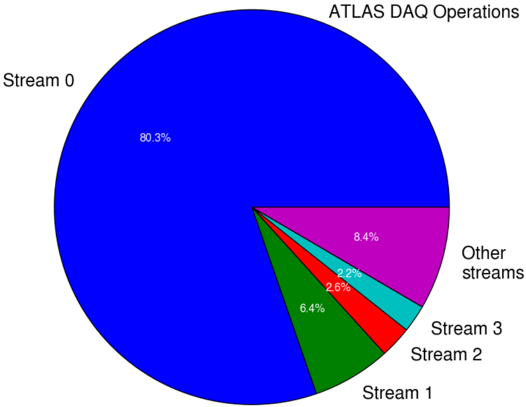


Figure: Bandwidth distribution between different streams for data collected on 28/10/2015. The four highest bandwidth streams are shown seperately and all other streams are summed together as "Other streams".

2016 Real-time Streams Writing Rate

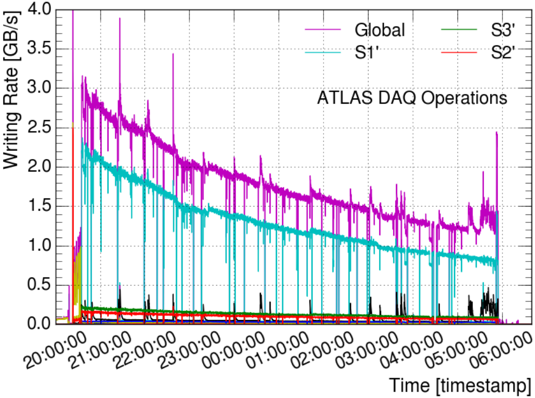


Figure: Instantaneous stream bandwidth for data collected on 24 and 25/10/2016. All streams are shown and each line represent a different stream. The highest line labeled "Global" is the sum of all streams representing the total bandwidth of selected events data.

2016 Stream Bandwidth Distribution

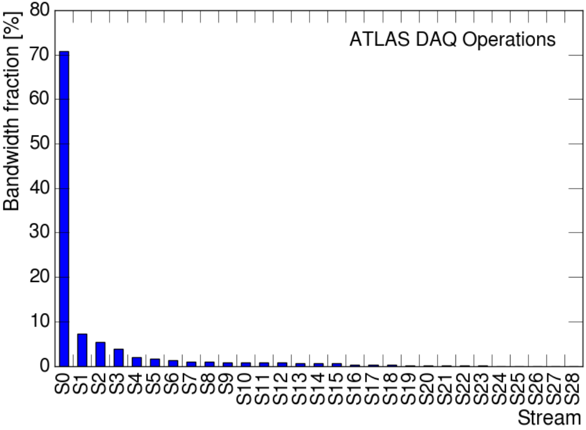


Figure: Stream bandwidth distribution for data collected on 24 and 25/10/2016. Each bar represent the fraction of the total bandwidth for one stream over the considered period.