Flexible event reconstruction software chains with the ALICE High-Level Trigger

D Ram\textsuperscript{1}, T Breitner\textsuperscript{1}, A Szostak\textsuperscript{2} for the ALICE HLT Collaboration
\textsuperscript{1}Johann-Wolfgang-Goethe University, Frankfurt, Germany
\textsuperscript{2}Institute for Physics, University of Bergen, Bergen, Norway
e-mail : dinesh.ram@cern.ch

Abstract : The ALICE High-Level Trigger (HLT) has a large high-performance computing cluster at CERN whose main objective is to perform real-time analysis on the data generated by the ALICE experiment and scale it down to at-most 4GB/sec - which is the current maximum mass-storage bandwidth available. Data-flow in this cluster is controlled by a custom designed software framework. It consists of a set of components which can communicate with each other via a common control interface. The software framework also supports the creation of different configurations based on the detectors participating in the HLT. These configurations define a logical data processing “chain” of detector data-analysis components. Data flows through this software chain in a pipelined fashion so that several events can be processed at the same time. An instance of such a chain can run and manage a few thousand physics analysis and data-flow components. The HLT software and the configuration scheme used in the 2011 heavy-ion runs of ALICE, has been discussed in this contribution.

1. Introduction

The ALICE High Level Trigger (HLT) is designed to perform real-time event analysis of heavy-ion and proton-proton collisions before they are sent to permanent storage. Computationally intensive analysis algorithms are performed in a large computing cluster. The HLT receives event data from all major detectors in ALICE. Processed results are sent back to the Data-acquisition system (DAQ).

Data read out from the front-end electronics (FEE) of the various detector systems in ALICE is received by the DAQ via optical fibers using a protocol called the Detector Data Link (DDL). The DAQ sends out an exact copy of the received data to the HLT farm via dedicated DDLs [1]. On the HLT side, this data is received by custom-designed PCI-X hardware - the HLT Readout Receiver Cards (H-RORCs). FPGAs on these cards pre-process the received...
data, execute cluster finding algorithms on the TPC data for example, before inserting the data into the main memory of their respective Front-End-Processors (FEPs) for the consumption of custom-designed software components, which perform additional analysis steps on the data before passing it on to the next component(s) in the hierarchy. The data-flow pattern in the cluster is defined in special configuration files which allow distribution of analysis tasks across the cluster.

Like in any large computing cluster evolving over time, the HLT cluster is quite heterogeneous in its hardware configuration. It consists of 2744 CPU cores, 64 GPUs and 246 FPGAs distributed over more than 200 multi processing nodes. The total amount of physical memory available in the cluster is approximately 5.3 TB. All the machines in the cluster are connected via a high-throughput InfiniBand network.

2. HLT Online Configuration

The synchronized and efficient transport of the data between individual processing components is managed by a custom-designed data transport framework[2]. The framework regulates the flow of data in the cluster in a pipelined fashion, which allows several events to be processed simultaneously in the HLT. The data-flow pattern emulates the physical layout of the detectors and independent software components are plugged together in a processing chain to achieve a specific analysis task. Each level in the data-flow hierarchy aims to reduce the data size by applying appropriate physics-analysis and reconstruction algorithms. Finally the data is made available in the form of Event-Summary-Data (ESD) for further offline analysis.

2.1 Configuration files

The HLT online configuration uses a set of configuration files written in XML format. These configuration files logically chain together all the data sources, data sinks and processing components necessary to achieve the physics analysis goals of the HLT. The scheme in which these files are written provides the necessary flexibility to plug-in new components to the analysis chain without any code modifications – and hence the name “Simple Chain Configuration”. Initially a configuration format called the SimpleChainConfig1 (SCC1) was developed which is generic for any software chain i.e. it was not specific to ALICE. The software chains running in ALICE use a different configuration format called the SimpleChainConfig2 (SCC2). This format is specific for ALICE and incorporates a lot of knowledge about the structure of ALICE and its detectors. The configuration files are very simple to read as they provide only a higher level picture of the flow of data within the cluster, specifying only the component names, the corresponding libraries, the nodes on which these components are to run and at what multiplicities. They abstract all the intermediate data-flow steps necessary to run the chain. Depending on the detectors sending data into the HLT and the analysis tasks desired, the software components can be chained together appropriately by modifying the SCC2 configuration files.
The SCC2 configuration files are compiled into low-level xml files which specify how the TaskManager control program[3] described in section 2.2, should start the processes in the chain and where. The program which compiles the SCC2 files, internally uses SCC1 data structures and algorithms to generate the low-level configuration files. In addition it also generates an SCC1 configuration file equivalent to the SCC2 configuration files as part of its output. The generated output files map each analysis component onto a machine on which it is supposed to run and define all the intermediate data-transport components, mostly involving scattering and merging of event data within each node as well as across nodes.

Three categories of components could be defined in the configuration files to make up a typical HLT software chain.

**Reconstruction components**: The HLT configuration in the 2011 heavy-ion run consisted of all the standard reconstruction components for the following detectors – Time Projection Chamber (TPC), Silicon Pixel and Strip detectors (SPD, SSD), the Electro-magnetic Calorimeter (EMCAL), the Photon Spectrometer (PHOS), The MUON detectors (trigger, and tracker), V0, and the Zero Degree Calorimeter (ZDC). After the individual first level detector reconstruction algorithms have processed the raw data, TPC tracks are prolonged with Inner

---

*Figure 1: Data-flow in the HLT. Data volume is reduced at each level of the hierarchy.*

---
Tracking Chamber (ITS) - SPD and SSD - space-points in order to create combined tracks and to determine the position of the primary vertex for each event. The produced tracks are then matched with clusters in the calorimeters (EMCAL and PHOS).

**Trigger Components**: There are one or more trigger components per detector and they all feed the global trigger component. A trigger decision is computed according to a specified trigger menu, by combining the results from all the other trigger components.

**Monitoring Components**: The monitoring components are responsible for producing the histograms which are used for live data quality monitoring and data visualization. These histograms are then sent to external systems to be monitored and archived. The HLT configuration is optimized for minimizing the latency of the software chain for it to be able to handle high data rates. At the moment, the entire configuration is defined by hand and process multiplicities are adjusted according to the expected data rates. Following the concept of locality of reference, components which are closely linked with each other in the data-flow logic are tried to be accommodated on the same machine as much as possible in order to avoid unnecessary intra-node communication and expensive gather and merge operations.

3. TaskManager Control System

The TaskManager control system implements a state-machine and provides the state-logic for the system as a whole. It defines a sequence of stable states that the system can be in, during the transition from an initial state to a running state and vice-versa. Controlled components have several states that they can be in. The TaskManager coordinates when the state-change commands are sent to each of these components and ensures that the entire system is in a consistent state by querying the components at regular intervals. If one or more of these components are in an erroneous or inconsistent state, the TaskManager control logs this inconsistency and it is also reflected in the overall state of the system.

Figure 2 illustrates a use-case diagram for the TaskManager Control system and its interaction with external systems like the ECS and the logging system. A brief description of the use-cases is provided below.

**Modify Configuration**: SCC2 configuration files are created or modified by a human expert and stored in a Subversion repository.

**Send Run Control Signals**: The Master TaskManager, with the help of a software interface known as the Experiment Control System (ECS) proxy, receives run-control commands for starting/stopping a run from the ECS. During the start of a run, the ECS sends a sequence of control strings to the HLT with information about the current run. The control strings contain the run number, the type of run, the beam type, the list of detectors participating in the run, the list of DDLs from which the HLT should expect data during this run and the trigger code.

**Control Slave TaskManagers**: The TaskManager control system is designed to effectively manage the thousands of processes distributed over a few hundred nodes. It follows a distributed management paradigm with one “slave” TaskManager process per node which are
in turn controlled by another “servant” TaskManager process one level up in the control hierarchy. The Master TaskManager is at the top of this control hierarchy and it is responsible for coordinating between all the other TaskManagers.

**Configure HLT chain**: The stored configurations are downloaded from the subversion repository and are compiled locally on the control node into low-level configuration files with the help of a suite of configuration tools, scripts and templates (containing state-change actions for all the TaskManagers). The generated low-level configuration files describe which processes are to be run and on which node. They are then copied over to each of the nodes defined in the configuration. TaskManager processes are started locally on every node by reading from these files. At this point, the HLT chain is said to be in a fully configured state.

**Start/Stop Chain**: The Master TaskManager then issues an “Engage” command to the already configured HLT chain. The “Engage” command triggers the spawning of individual analysis and data-transport processes on each node by reading in binaries from copies on local discs. At this point, the chain is said to be “Running” and ready to receive data from the detectors.

**Manage Components**: TaskManagers continuously query the state of each of the processes that it controls directly and in turn reports this information to the TaskManager processes, one
level up in the control hierarchy.

**Log error** : A TaskManager process is also responsible for reporting and logging error conditions of any of the components it controls. It also uses external logging libraries to record this error into a database.

**Show error** : The reported errors are accessed by human operators and experts and are handled appropriately.

4. HLT Online Chain Running in 2011

The average data rate read out by all detectors during the 2011 heavy-ion runs was around 7 GB/sec, which means the HLT chain was receiving 7 GB of raw data per second as input. The average raw data size per event for all detectors was around 30 MB of which, the TPC was by far the largest contributor to the data volume read out by ALICE with the average TPC event size around 25 MB/event. To handle such high data rates, each instance of the processing chain approximately runs 40-50 different physics analysis components, most of them running several instances in parallel for load balancing purposes, amounting to a total of approximately 1600-1800 processing instances and 9000-10000 data-flow instances, distributed over the cluster nodes. Each analysis process instance is roughly assigned one core and 2 GB of physical memory.

![Configuration size vs. Time](image)

*Figure 3 : Configuration size vs. Time*
The volume of data handled by the HLT during the heavy-ion period of ALICE in 2011 was significantly higher than in 2010, as expected. To cope up with the increased data rates, the size of the HLT online configuration, in terms of the total number of processes running in one instance of the online chain also increased. As illustrated in Figure 3, as the size of the configuration increases, so does the time required to start-up the chain, following the sequence as described in section 2.1. In this figure, data relating to several configurations generated during actual runs are shown. An optimum performance by the file system and the network is assumed.

The size of the configuration depends on the combination of detectors participating in each run, the detector links from which the HLT is receiving data and the analysis tasks chosen for that run.

5. Current Issues and improvements

In the current setup, all the individual detector reconstruction chains are part of a single HLT software chain. If any of the components fail during the run, data taking has to be stopped. The crash of a component could be for several reasons – a memory leak or the lack of resilience of a component to handle corrupt data coming from the detector. It is natural that in a big experiment like ALICE, evolving physics goals necessitate several hardware and software changes during the lifetime of the experiment. As a result, the analysis components undergo a constant development cycle. As more code is added, the chance that it introduces a new bug or exposes a previously unknown bug is also quite high. Several of the crashes are unfortunately detected only during actual running and might not be reproducible offline. The HLT software chain could also fail due to hardware problems encountered during a run – a failed node for example, which needs to be replaced. However, as it is shown in [4], the HLT is much more stable now than in the last year.

5.1 Improvements

Restarting crashed components: A crashed process could be restarted a certain number of times before bringing down the entire chain. The data-transport framework supports buffering of data-streams. So, in principle, partially processed event fragments could be buffered and that section of the chain temporarily stalled, while a new process is created to replace the crashed component and the traffic rerouted through this newly created process. The TaskManager code already contains logic to do this, but it has not been fully implemented and tested. This approach could be extended in order to run several independent chains which feed to a main control chain, so that when one chain crashes it could be independently and automatically restarted without bringing the entire system down. This would however require major changes in the existing software.

Critical and non-critical process classification: Processes in the chain can be classified as critical or non-critical. A critical process is one which is essential for the reconstruction to proceed normally – the track matching component for example – whereas, a histograming component is non-critical for the run the continue. So, a separate chain can be run independent of the main chain for such components. This solution is already being followed since the 2011 heavy-ion runs, where several of the monitoring components are run in an independent chain.
6. Conclusion

In its present state, the software and the computing cluster are able to handle the volume of data generated by ALICE comfortably. The maximum data rate that was observed during actual runs was limited to 1.5 Khz in pp and 200 Hz for most central collisions in Pb-Pb. However, in future, ALICE is expected to cope with significantly higher interaction rates, which would be possible after the proposed upgrades in the different detector and data acquisition systems. This would also significantly increase the volume of data being generated by ALICE. The HLT is expected to host more intensive and complicated triggering and data compression applications to be able to handle those rates.

As more detectors participate in the HLT and with the increasing data challenges posed by ALICE, from the computing point of view, it translates into a need to efficiently manage an even higher number of software components communicating with each other and competing for the same resources in the cluster. An upgrade strategy for the HLT hardware and modifications to the software and control system is currently under discussion.

References

[1] ALICE Collaboration, ALICE Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System, ALICE-TDR-10, CERN-LHCC-2003-062 (2003)

[2] Steinbeck T M et al 2002 An object-oriented network-transparent data transportation framework, IEEE Trans. Nucl. Sci. 49 (2002)

[3] T. Steinbeck et al., “A control software for the ALICE high level trigger,” in Proc. Computing in High Energy Physics Conf. 2004 (CHEP04), 2004 [Online].