ALICE High Level Trigger Interfaces and Data Organisation

Sebastian Robert Bablok*, Matthias Richter, Dieter Roehrich, Kjetil Ullaland
Department of Physics and Technology, University of Bergen, Norway
Torsten Alt, Volker Lindenstruth, Timm M. Steinbeck, Heinz Tilsner
Kirchhoff Institute of Physics, Ruprecht-Karls-University Heidelberg, Germany
Harald Appelshaeuser
Institute for Nuclear Physics, University of Frankfurt, Germany
Haavard Helstrup
Faculty of Engineering, Bergen University College, Norway
Bernhard Skaali, Thomas Vik
Department of Physics, University of Oslo, Norway
Cvetan Cheshkov
CERN, Geneva, Switzerland
for the ALICE Collaboration

Abstract

The High Level Trigger (HLT), integrating all major detectors of the heavy-ion experiment ALICE, is designed to analyse Large Hadron Collider (LHC) events online. A cluster of 400 to 500 dual SMP PCs will constitute the heart of the HLT system. To synchronise the HLT with the other online systems of ALICE (Data Acquisition (DAQ), Detector Control System (DCS) and Trigger (TRG)), the Experiment Control System (ECS) has to be interfaced. In order to do so, an implementation of Finite State Machines (FSM), with the usage of SMI++, offers an easy way to coordinate the running conditions of the HLT with the other online systems. The mapping of the HLT states has to offer the possibility to run the HLT in stand alone mode (for commissioning and calibration) as well as controlled by ECS during the operational stage.[1]

After a full reconstruction of an event, the HLT provides trigger decisions, Regions-of-Interest (RoI), and compressed data to the DAQ in order to reduce the data rate to permanent storage. In addition, the result of the online event reconstruction, organised as Event Summary Data (ESD), has to be classified and indexed for later reference to the offline analysis.

For optimised usage of the computing power of the HLT cluster, it is foreseen to interface it to the GRID middleware AliEn (Alice Environment).

OVERVIEW

ALICE is one of the experiments at the new LHC at CERN. Although the detector can be operated in proton-proton (pp) mode, it is mainly designed for heavy-ion collisions. In the latter case large multiplicities with up to 15.000 particles per event are expected with an event size of up to 80 MB and a rate of 200Hz for the TPC (Time Projection Chamber), the sub-detector which will deliver by far the largest data volume. For pp mode the data rate will be somewhat lower.

As its main purpose, the HLT in ALICE will provide a fast final trigger decision to the DAQ. In addition, preprocessed and compressed data will be offered (lossless data reduction as far as possible). Therefore, events will be filtered, RoIs selected and events (partially) reconstructed. The HLT will also be able to offer a first performance impression of the ALICE detector when the LHC is started in 2007. [1]

The HLT will consist of a big PC cluster with up to 500 nodes, equipped with dual CPU SMP boards and most likely dual core CPUs. This leads to a sum of up to 2000 available processing units. The nodes will at least be connected via Gigabit Ethernet. Therefore, the HLT is comparable to a GRID Tier 2 center. Up to 300 of the nodes will be equipped with HLT-ReadOut Receiver Cards (H-RORC) to receive copies of the event data from the DAQ-ReadOut Receiver Cards (D-RORC). These nodes are called Front-End-Processing (FEP) nodes. On the H-RORCs and FEPs the data will be preprocessed and then distributed in the cluster via a Publish-Subscriber (PubSub) system for further processing. [2]

The cluster nodes will be arranged in multiple hierarchy levels matching the detector layout as well as the sequence of analysis steps required. The organisation of the nodes will be handled by a complex system of TaskManagers, running on each node. A set of multiple distributed master TaskManagers, configuring and controlling the cluster, will avoid single-points-of-failure and allow a fail-safe operation of the cluster. [2]

The HLT can handle an input data stream of up to 25 GB/s. After processing in various stages the data will be returned to the Local Data Concentrators (LDCs) of the DAQ with an output stream of at most 1.2 GB/s.[1]

Interfaces to the ECS, the DCS, the Offline system (AliRoot library), and the DAQ provide for a homogeneous integration into the other online systems of ALICE. In addition, it is planned to have an interface to AliEn, the GRID

* Sebastian.Bablok@ift.uib.no

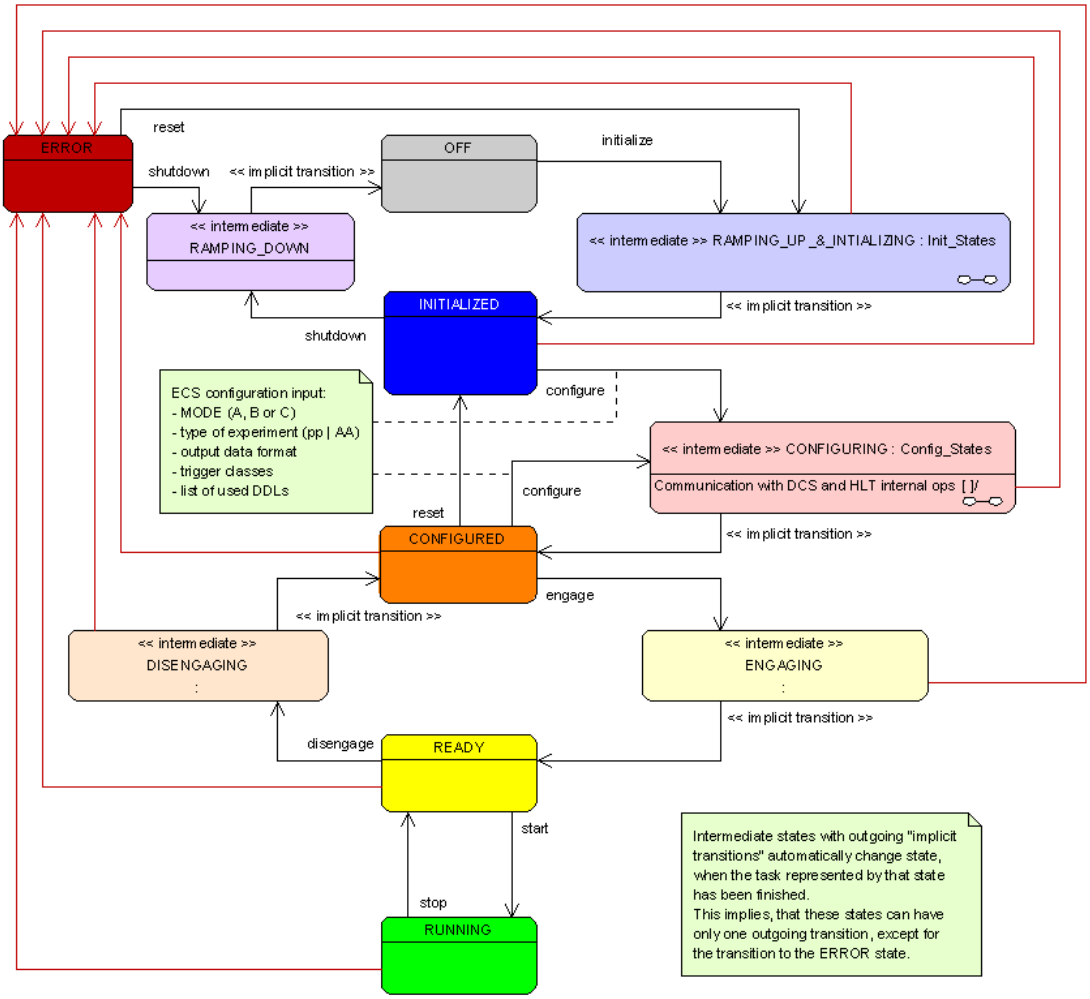

Figure 1: State diagram representing the interface between the ECS and the HLT

middleware for ALICE, to offer the computing power of temporarily unneeded nodes.

HLT - ECS INTERFACE

HLT States

The interface between the HLT and the ECS will be represented as a set of well defined states of a Finite State Machine (FSM), the HLT proxy. These states mainly reflect the complex initialisation and configuration procedures of the HLT cluster before reaching running conditions. Due to the fact that some of these steps may take quite some time, intermediate states have been introduced to signal the process of these preparation tasks.

The ECS will request the current state from the HLT proxy and issue commands for state transitions. Fig. 1 displays all possible states of the HLT proxy and their transitions.

Transitions and Commands

Normally each state transition requires an explicit command to the HLT proxy. This is different for the intermediate states, which will transition automatically to the next "stable" state when their preparation processes have finished. This type of transition is called an *implicit transition*. The *ERROR* state is the only exception to this rule. It can be reached from all states without any command. This will occur in cases of failures, which lead to incorrect operations of the cluster (e.g. too many FEP nodes failed, too many cluster nodes down and not recoverable, all multiple master TaskManagers are unreachable).

For correct operation the HLT cluster needs additional information about the current running conditions. They will be given as parameters to the *configure* command:

- **operating mode** of the HLT in the current run:
  A: DAQ mode - no HLT functionality in the system
  B: HLT processing, but DAQ ignores trigger decision
  C: full HLT functionality - trigger and data processing
- **type of run**: proton-proton or heavy-ion collision

- **output format**: version of the output data format transfered to the DAQ system
- **trigger classes**: list of tags identifying the desired physic triggers
- **list of DDLs**: list of used DDLs for HLT data input

Internal Communication

Internal communication between the HLT proxy and the master TaskManagers will be performed via communication libraries of the TaskManager control system. The TaskManagers can send interrupts to the HLT proxy to notify the proxy of state changes in the cluster. The HLT proxy will take a majority vote of the master TaskManagers.

At start up, the HLT proxy will load the dedicated interface library as a shared library, which wraps the actual (network-) communication. It allows for automatic adaption to the flexible internal cluster configuration.[2] [3]

Implementation

The implementation uses SMI++ (State Management Interface), which is common among all other system connected to the ECS. In general, SMI++ allows for partitioning and multiple instances of one domain, but from the point of the ECS the HLT cluster will represent itself as a single instance.

SMI++ is developed at CERN and uses DIM (Distributed Information Management) as communication framework to exchange data between FSMs and proxies across different domains [4] [5]. Fig. 2 shows the different communication mechanism used in the HLT cluster control.

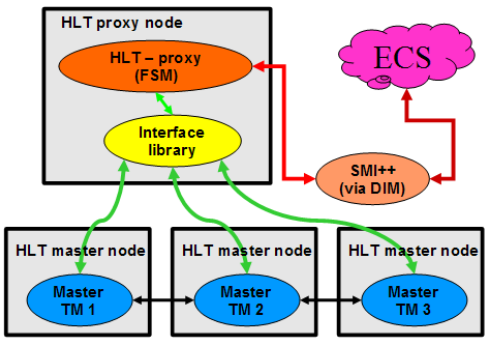

Figure 2: Overview of the communication in the HLT cluster control.

The internal distribution and control of tasks and partitioning of duties in the HLT cluster will be managed by the TaskManager system and is described together with the HLT data transport framework (PubSub) in former CHEP proceedings [2] [3] [6].

The HLT proxy integrates into the existing software for the internal cluster control and adopts to their requirements of:

- **flexibility**: The TaskManagers can dynamically optimize the internal cluster settings according to the currently desired analysis procedure steps during the configuration phase without interference of the HLT proxy.
- **hierarchy**: The HLT proxy contacts the multiple master TaskManagers and acts for them as master.
- **fault tolerance**: Using a majority vote of the different master TaskManagers will avoid single-points-of-failure (failing of the HLT proxy once the running state has been reached, will not affect the cluster itself or its tasks, event data will still be processed and delivered to the LDCs of the DAQ system).

The HLT proxy is only sensitive for the system during the initial ramping up and configuration period, which is a short time compared to the running state. The main aim is to avoid data loss during run time. This is mainly handled by the fault tolerant mechanisms of the TaskManager system.[3]

HLT - DAQ DATA FORMAT

After processing event data and making trigger decisions the HLT will deliver event data back to the DAQ. In this context the DAQ handles the HLT just like any other detector: It will receive the data via D-RORCs in serveral of its LDCs.

When running in mode *"C"* the HLT delivers trigger decisions as well as preprocessed and analysed event data to the DAQ. In case of an accept the HLT sets up a list of DDLs defining the RoI for this event. The DAQ should write all event data received from the selected DDLs of the detectors to permanent storage. In addition, DAQ has also to store the preprocessed and compressed data of the HLT. In mode *"B"*, DAQ will not evaluate trigger decisions from the HLT. But the full HLT processing will be performed nonetheless and data will be delivered to DAQ for storage. For performance analysis of the HLT the trigger decisions (ACCEPT and NOT ACCEPT) should be saved as well. In mode *"A"* the HLT will provide no data output to the DAQ.

In detail the proposed output format for data from the HLT to the DAQ will include the following:

- **trigger decision**:
  - list of DDLs to read (detector and HLT DDLs)
- **event summary data (ESD)** [in case of an accept]:
  - candidate ID: ID from trigger class
- **preprocessed event data** [optional]:
  - (partially) reconstructed events
  - compressed raw data
  - preprocessed detector specific data (e.g. Photon Spectrometer pulse shape analysis)
- **special events** [optional]:
  - start of data (SOD): configuration
  - end of data (EOD): run summary

HTL CLUSTER FOR GRID (ALIEN)

GRID Portal

As stated before, the HLT cluster is similar to a Tier 2 center with up to 2000 processing units. When not performing HLT tasks, it is foreseen to use them for GRID computing. For this purpose, a dedicated node will act as GRID portal. This node will have installed the ALICE GRID middleware AliEn. It will be a Computing Element in AliEn but no Storage Element, at least no permanent one because of the priority on HLT tasks.

The portal node will have contact to the AliEn system, handle the needed certificates and authentication methods and accept GRID jobs. It will be able to include nodes into the GRID, providing an estimated Time-To-Live (TTL) for each included node. The TTL announces how long this node is expected to be available for GRID. An algorithm, which will develop according to the experiences gathered with the HLT cluster, will calculate the TTLs. The portal can also exclude nodes from the GRID when the HLT demands more computing power. Accepted GRID jobs will be distributed in the HLT cluster using the batch system Condor [7]. Therefore Condor has to be installed on all nodes as well. Application binaries for AliEn will be shared via network among all participating nodes.

The usage of the HLT cluster can be divided into two periods: ALICE data taking periods, when only nodes, unused by the HLT, will be offered to AliEn, and non-data taking times, when the whole cluster can be offered for GRID computing. The priority for the HLT cluster will be an undisturbed execution of HLT tasks. In order to do so, the portal node will keep contact with the HLT proxy and the resource management system of the cluster. It has also to administer the pool of unused nodes.

GRID Pool Administration

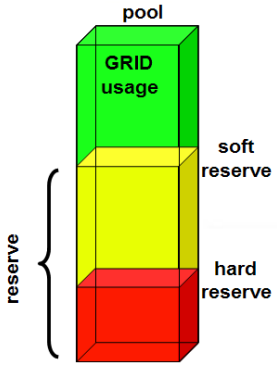

Figure 3: The buffered pool of unused cluster nodes.

All nodes currently not needed for the HLT will be collected into a buffered pool. This pool will have three regions separated by two thresholds. Nodes, which are needed for the HLT will be taken immediately from the reserve of the pool. If the amount of nodes in the pool exceeds a certain limit, any further node entering the pool will be offered for GRID computing. This region is called *GRID usage*. The other two regions together form the pool *reserve*. If the amount of nodes in the pool is below the *soft reserve* (the upper threshold), no more nodes are offered for GRID. Unused nodes from the HLT will be only collected in the reserve. Nodes finishing AliEn jobs will also be recollected into the reserve, performing a *"soft" stop*. The third region forms some kind of safety minimum for the HLT. If the amount of nodes inside the pool falls below the *hard reserve* (the lower threshold), GRID nodes are immediately recollected to the reserve, performing a *"hard" stop*, until the hard reserve is exceeded again. This implies that currently running GRID jobs on these nodes will be canceled. The aim of the hard reserve is to have spare nodes immediately available for the HLT. The soft reserve has been introduced to avoid "hard" stops of GRID nodes as often as possible. Fig. 3 illustrates all pool regions.

In this context, the non-data taking periods are just a special case of the normal pool administration.

SUMMARY AND OUTLOOK

The HLT will consist of a large PC cluster, similar to a Tier 2 center. Internally it will be managed by a TaskManager control system. Overall system configuration and control will be handled by the ECS using well defined states and transitions. Implementation of the interface will be in SMI++. The HLT will provide trigger decisions as well as (pre-) processed data to the DAQ. An interface to AliEn will allow for GRID computing in the cluster, using Condor as an internal batch system. A buffered pool will take care of the node distribution between HLT and GRID tasks.

Taken together this allows to utilise the computing power of the whole HLT for multiple applications.

ACKNOWLEDGEMENTS

Work on the ALICE HLT has been supported by the Norwegian Research Council (NFR).

REFERENCES

[1] ALICE Technical Design Report 010, Printed at CERN, CERN-LHCC-2003-062, ISBN: 92-9083-217-7, 2003

[2] T. M. Steinbeck, V. Lindenstruth, H. Tilsner, "New Experiences with the ALICE High Level Trigger Data Transport Framework", CHEP04, Interlaken, Switzerland, 2004.

[3] T. M. Steinbeck, V. Lindenstruth, H. Tilsner, "A Control Software for the ALICE High Level Trigger", CHEP04, Interlaken, Switzerland, 2004.

[4] http://smi.web.cern.ch/smi/

[5] http://dim.web.cern.ch/dim/

[6] T. M. Steinbeck, V. Lindenstruth, H. Tilsner, "A Software Data Transport Framework for Trigger Applications on Clusters", CHEP03, La Jolla, USA, 2003.

[7] http://www.cs.wisc.edu/condor/