New Experiences with the ALICE High Level Trigger Data Transport Framework

Timm M. Steinbeck*, Volker Lindenstruth, Heinz Tilsner,
Kirchhoff Institute of Physics, Ruprecht-Karls-University Heidelberg, Germany,
for the ALICE Collaboration

Abstract

The ALICE High Level Trigger (HLT) is foreseen to consist of a cluster of 400 to 500 dual SMP PCs at the start-up of the experiment. Its input data rate can be up to 25 GB/s. This has to be reduced to at most 1.2 GB/s before the data is sent to DAQ through event selection, filtering, and data compression. For these processing purposes, the data is passed through the cluster in several stages and groups for successive merging until, at the last stage, fully processed complete events are available. For the transport of the data through the stages of the cluster, a software framework is being developed consisting of multiple components. These components can be connected via a common interface to form complex configurations that define the data flow in the cluster. For the framework, new benchmark results are available as well as experience from tests and data challenges run in Heidelberg. In addition the framework has been used during testbeam experiments of an ALICE TPC prototype chamber.

OVERVIEW

ALICE [1] [2] [3] is A Large Ion Collider Experiment for the future Large Hadron Collider (LHC) being built at CERN. It will take data in different modes of LHC operation, in pp as well as in heavy-ion collisions. From the point of view of its High Level Trigger (HLT) [4] system among the most important characteristics in heavy-ion mode are the maximum event size up to 75 MB. Together with the event rates from the different detectors, e.g. up to 200 Hz for the TPC, this defines the requirements for the data stream that the HLT has to handle. In pp mode the requirements on the data stream are somewhat lower with a maximum TPC rate of up to 1 kHz and an event size of about 2.5 MB taking into account the pile-up of 20 pp events. But in this mode the higher event rates from the detectors are emphasised which also have to be handled by the system.

ALICE HLT's primary task is the reduction of its input data stream of up to 25 GB/s to at most 1.2 GB/s, which are accepted by the data acquisition for storage to tape. For this purpose three basic measures are taken by the trigger system: filtering of interesting events, selection of regions-of-interest in events, and online compression of the filtered and selected event data. In order to fulfill its purpose the HLT performs a full online event reconstruction from the raw data of the detectors participating in the trigger decision. A PC cluster of initially 400 to 500 dual CPU PC nodes will be used to perform the analysis. The cluster nodes will be arranged in multiple hierarchy levels matching the detector layout as well as the sequence of analysis steps required.

* timm.steinbeck@kip.uni-heidelberg.de

THE DATA TRANSPORT FRAMEWORK

As the exact processing sequence and hierarchy for the HLT are not yet fixed a flexible and efficient software framework [4] [5] [6] has been developed to transport the event data through the cluster. It consists of multiple independant components communicating via a fixed interface according to the publisher-subscriber or producer-consumer principle. Named pipes are used to exchange small messages and descriptors between the components, while shared memory is used for the event data itself, reducing the amount of copying being done to a minimum.

In addition to a number of finished and ready-to-use data-flow components templates are provided for application specific components. Components for different tasks can be easily built using these templates: Obtaining data and inserting it into a component chain (data source template); accessing input data, processing it and producing some new output data (data processing template); accessing input data and providing data to entities outside of a component chain (data sink template). Existing data flow components provide means to merge different parts belonging to the same event (Event Merger), split and rejoin a stream of events, e.g. for load-balancing, (Event Scatterer & Gatherer), and to send event data across a network between different nodes (Subscriber & Publisher Bridge Heads).

BENCHMARKS AND TESTS

Interface Benchmarks

In order to determine the performance and scaling behaviour of the current framework interface versions and expand the existing data set the publisher-subscriber interface has been evaluated on a number of different PCs. The evaluation consists of benchmark publisher and subscriber components. In the benchmark the publisher announces events in succession as quickly as possible up to a specified limit of events that are in transit at any time. The subscriber receives these events and immediately sends an event release message back to the subscriber. Neither of these two components actually places or retrieves any data in shared memory or dereferences descriptors to shared memory locations. Only the exchange of descriptors and messages is measured by this test.

Results of the test are shown in table 1. All tests have been run under Linux, with different kernel versions. On some of the dual SMP PCs the tests have been run with a single CPU kernel in addition to the SMP kernel. The error in the SPECint rating in the table approximately reflects the variations between the benchmarked configurations and those present in the SPECint result list. Among the most striking things in the benchmark measurements are the poor results from the whole Pentium 4 family (Celeron, Pentium, and Xeon). These results can most likely be explained by the family's level 1 (L1) cache size and the fact that the components use two named pipes to communicate, one each from publisher to subscriber and vice-versa. As each of these pipes uses a memory area of 4 kB in the kernel the CPU's 8 kB L1 data cache is quickly filled. This filling then leads to a lot of cache thrashing as data pieces are quickly replaced in the cache before being reused often enough. In contrast one can conclude that the AMD Athlon and Opteron architectures profit from their large L1 data caches of 64 kB each. Here the Opteron seems to super-scale compared to the Athlon simply based on clock frequency, which can most likely be attributed to the Opterons' improved internal architecture as well as its larger L2 cache size of 1 MB compared to 512 kB. In [4] the prediction was made that by the time ALICE and the HLT start to operate the CPU overhead for a complete event-announce-finished loop, as measured in this benchmark, would be at most 15 μs. Looking at the results of the Opteron measurement and taking into account the two CPUs involved, one can see that already now less than 18 μs CPU overhead per event announcement are used. As this number can be expected to decrease further one can conclude that the CPU overhead imposed by the framework interface will reach the predicted 15 μs and be small when ALICE and the HLT start to operate.

One should note, that the cache effects observed in these benchmarks are somewhat of an artifact. They are unlikely to be visible to such an extent in a real system, where analysis code is processing the events. In such a scenario significantly more CPU time will be used for analysis than for the transport of the data in the framework, and a high fraction of the CPU caches will be taken up by the event data, rather than the framework descriptors. Nonetheless the benchmarks show that the framework itself is efficient and produces only a small overhead for the transport of the data on one node. Furthermore, burst announcements effects can be expected to appear in a complex system, mainly due to timeslicing of CPUs to processes in multi-tasking operating systems. This burst pattern is expected to produce effects of caching similar to those seen in these benchmarks. Significant numbers of events can then be announced in bursts in quick succession, supporting good cache reuse as well.

HLT Data Challenges

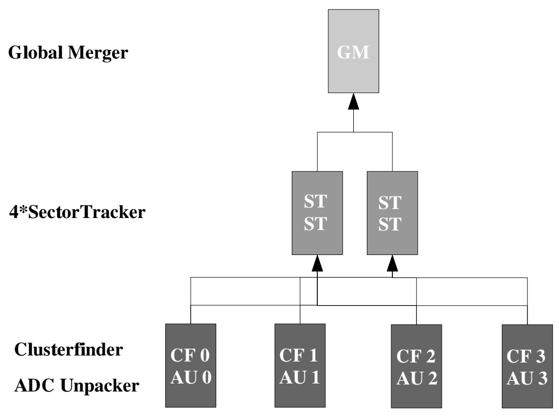

Figure 1: Test Setup for the HLT Data Challenge

As a test of the stability of the framework when it is used in a more complex systems a data challenge has been setup on a small cluster in Heidelberg. The part of the cluster available for the test consists of seven dual Pentium 3 nodes with 800 MHz CPUs and 512 MB of memory each. In the test a setup simulating a reduced TPC sector readout has been used as displayed in Fig. 1. Four instead of six readout PCs were used at the start of the chain and only two nodes for sector level tracking and one for data merging. Since the data challenge was intended as a stability test of the data transport framework and not of any analysis code, no actual processing components have been used. Instead the analysis components have been replaced by dummy analysis components that just copy a part of their input data into their output area. The amounts of data to be copied have been chosen to resemble the relative data sizes produced by the corresponding real analysis components. Two dummy analysis components were used on six of the PCs to utilise both CPUs, for a total of 13 analysis components. Four source components are used to insert test data from files into the component processing chain. A final sink component is used to measure the event rate achieved. In addition to these application components 39 data flow components are used throughout the system. For each data flow component type listed in the framework section above at least one instance has been used. This ensures that each type of data flow component will be exercised during the test. After activation the described setup has been running continously for a month before being stopped explicitly without error condition. During the running time more than 2×10^9 events have been transferred through the system, with an overall average event rate of more than 780 Hz.

TPC Sector Beamtest

Beyond the local data challenges in Heidelberg the HLT was also used during the beamtest of a TPC sector prototype at CERN in May/June 2004. The HLT was not used to

Table 1: Interface Benchmarks

CPU(s)	**Chipset**	**Kernel Version**	**SPECint Rating (approx.)**	**Event Rate / kHz**
Dual Pentium 3 800 MHz	Serverworks HEsl	2.4.23	700 ± 30	13.7 ± 0.03
Dual Pentium 3 800 MHz	Serverworks HEsl	2.4.23 / 1 CPU	350 ± 10	11.1 ± 0.06
Dual Pentium 3 733 MHz	Serverworks HE	2.4.23	660 ± 30	11.9 ± 0.2
Dual Pentium 3 733 MHz	Serverworks HE	2.4.23 / 1 CPU	330 ± 10	10.1 ± 0.1
Celeron 4 1.7 GHz	Intel 845	2.4.22	550 ± 100	9.2 ± 0.08
Pentium 4 2.6 GHz w. HyperThreading	Intel 865	2.4.22	1100 ± 10	17.6 ± 0.1
Pentium 4 2.6 GHz no HyperThreading	Intel 865	2.4.22	1000 ± 30	18.9 ± 0.8
Dual Xeon 4 2.8 GHz w. HyperThreading	Unknown	2.4.20	2400 ± 30	20.5 ± 0.005
Athlon 600 MHz	AMD 750	SuSE 2.4.20	300 ± 100	24.6 ± 0.1
Dual Opteron 248 2.2 GHz	AMD 8XXX	2.4.20-8	2700 ± 30	117.2 ± 0.2

provide online data analysis capability for this test. Instead the aims of the participation were to gain some experience in using the HLT in a real experimental situation and to test the functionality and correctness of the hardware interface between the HLT and DAQ systems [4].

This interface uses the ALICE Detector Data Link (DDL) optical links between DAQ and HLT. Detectors are readout by the DDL into PCI DAQ Read-Out Receiver Cards (D-RORCs) in DAQ front-end PCs, Local Data Concentrators (LDCs). Splitters on the LDC D-RORCs transmit the data via identical DDLs to PCI HLT Read-Out Receiver Cards in HLT Front-End-Processor PCs (FEPs). The HLT-RORCs place the received data into the PCs main memory to be retrieved by dedicated framework source components. At the end of a HLT component chain a sink component retrieves the HLT output data's address and feeds it to another PCI card, the HLT Output card (HLT-Out). The HLT-Out then retrieves this data from the computer's main memory and sends it again via DDL to a D-RORC in a DAQ-LDC.

In the setup that was intended to be used for the beamtest three DAQ and HLT PCs each were involved. One DAQ LDC received data from the detector prototype via two DDL links and two D-RORC adapters. Each of the DDLs was forwarded via the D-RORC splitter to a HLT-RORC in a separate HLT FEP. Data from the HLT FEPs is sent via TCP over Gigabit Ethernet to a third HLT node. It merges the data and sends it to a second DAQ LDC, using an HLT-Out, DDL, D-RORC sequence. Both DAQ LDCs send their received data to the third DAQ PC, a Global Data Concentrator (GDC). Finally, the GDC merges the event data and transmits it to permanent storage.

During the activation of the setup problems occured which were traced to the 2 MB data size limit in the protocol of the DDL. In order to work around the problem the HLT setup used was decreased from two receiving links and FEPs to one. In the HLT setup actually used there have therefore been only two PCs in total, one receiving data from the detector via DAQ and one sending data to DAQ. Between these two HLT PCs data was transferred as planned using TCP over Gigabit Ethernet. Another problem related to the trigger of the testbeam setup caused the input data rate to remain below 5 Hz during the time of HLT activity. Regarding the achieved rates the HLT was therefore not put under stress.

The software setup used on the three/two HLT PCs is shown in Fig.2. In the final setup one RORCPublisher source component was used to access the data written into main memory by the HLT-RORC. This data was written to the PC's local disk by a StorageWriter sink component as well as sent over the network to the second active PC. On the second PC two sink components determine the rate at which event data is received and feed the received data to the HLT-Out card in the PC for transmission to DAQ as detailed above.

During the test both the DAQ-HLT interface as well as the HLT software components functioned well. Data could be received from the D-RORC's splitter and sent back to the second D-RORC without problems. The software also worked as expected without difficulties during operation.

SUMMARY AND OUTLOOK

From the results presented in this paper one can see that the ALICE High Level Trigger Data Transport Framework is already quite mature. Even on relatively old hardware it shows performance that is good enough for use in heavy-ion mode of ALICE and can be expected to be fast enough for proton-proton mode on newer hardware. CPU overhead due to the publisher-subscriber interface has been measured to be low enough even on contemporary hardware and should be even lower by the time ALICE starts to operate. The data challenge in Heidelberg with its one month of continous running time has demonstrated that the framework has reached significant stability. The component ap-

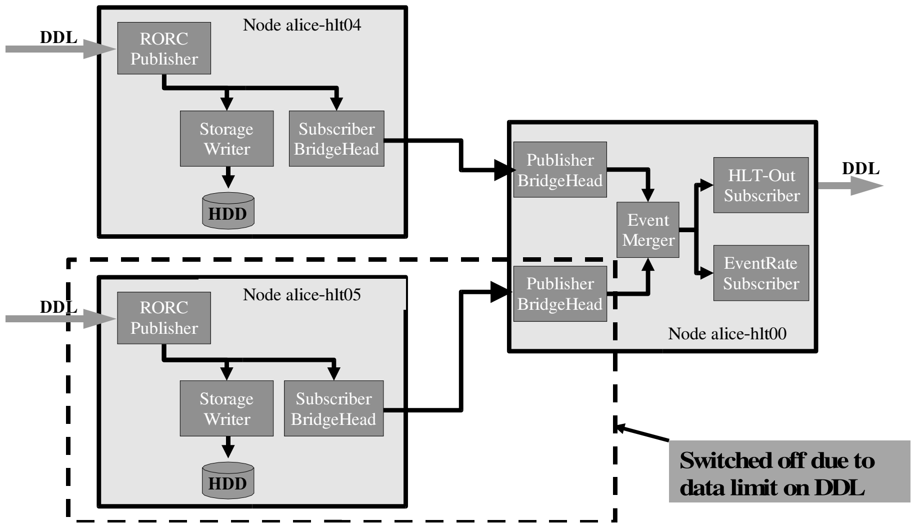

Figure 2: HLT Setup for the TPC Sector Beamtest

proach chosen for the framework has proven to be useful both for short tests as well as during the beamtest participation, because of the abilities to quickly create small configurations and easily adapt configurations to changing requirements. In the future more performance tuning will be attempted as well as work on integration with the TaskManager control system [7]. The focus there will lie in particular on fault tolerance capabilities, making use amongst others of the framework interface's dynamic connection ability. The framework together with auxiliary packages is available from [8], documentation can be found at [9].

ACKNOWLEDGEMENTS

Work on the ALICE High Level Trigger has been financed by the German Federal Ministry of Education and Research (BMBF) as part of its program “Förderschwerpunkt Hadronen- und Kernphysik - Großgeräte der physikalischen Grundlagenforschung”.

REFERENCES

[1] http://ALICE.web.cern.ch/ALICE/.

[2] http://ALICE.web.cern.ch/ALICE/user.html.

[3] The ALICE Collaboration, “ALICE - Technical Proposal for A Large Ion Collider Experiment at the CERN LHC”, CERN/LHCC/95-71, LHCC/P3, December 1995.

[4] The ALICE Collaboration, “ALICE - Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger, and Control System”, CERN/LHCC/2003-062, January 2004.

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

[6] T. M. Steinbeck et al., “A Framework for Building Distributed Data Flow Chains in Clusters”, Lecture Notes in Computer Science LNCS 2367, Proceedings of the 6th International Conference on Applied Parallel Computing, PARA 2002, Springer Verlag Berlin Heidelberg, 2002.

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

[8] http://www.ti.uni-hd.de/HLT/software/software.html

[9] http://www.ti.uni-hd.de/HLT/documentation/documentation.html