Commissioning of the ALICE High-Level Trigger

Dissertation

zur Erlangung des Doktorgrades

der Naturwissenschaften

vorgelegt beim Fachbereich Physik

der Johann Wolfgang Goethe-Universität

in Frankfurt am Main

von

Jochen Thäder

aus Heidelberg

Frankfurt 2012

(D30)

vom Fachbreich Physik der

Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan : Prof. Dr. Michael Huth

Gutachter : Prof. Dr. Harald Appelshäuser

Prof. Dr. Volker Lindenstruth

Datum der Disputation : 31.10.2012

Quand tu veux construire un bateau,
ne commence pas par rassembler du bois,
couper des planches et distribuer du
travail, mais reveille au sein des
hommes le desir de la mer grande et large.

Citation attribuée a Antoine de Saint-Exupéry

Durch Menschen bewegen sich Ideen fort,
während sie in Kunstwerken erstarren
und schließlich zurückbleiben.

Joseph Beuys

Abstract

A new era in experimental nuclear physics has begun with the start-up of the Large Hadron Collider at CERN and its dedicated heavy-ion detector system ALICE. Measuring the highest energy density ever produced in nucleus-nucleus collisions, the detector has been designed to study the properties of the created hot and dense medium, assumed to be a Quark-Gluon Plasma.

Comprised of 18 high granularity sub-detectors, ALICE delivers data from a few million electronic channels of proton-proton and heavy-ion collisions. The produced data volume can reach up to 26 GByte/s for central Pb–Pb collisions at design luminosity of $\mathscr{L} = 10^{27} cm^{-2} s^{-1}$, challenging not only the data storage, but also the physics analysis. A High-Level Trigger (HLT) has been built and commissioned to reduce that amount of data to a storable value prior to archiving with the means of data filtering and compression without the loss of physics information. Implemented as a large high performance compute cluster, the HLT is able to perform a full reconstruction of all events at the time of data-taking, which allows to trigger, based on the information of a complete event. Rare physics probes, with high transverse momentum, can be identified and selected to enhance the overall physics reach of the experiment.

The commissioning of the HLT is at the center of this thesis. Being deeply embedded in the ALICE data path and, therefore, interfacing all other ALICE subsystems, this commissioning imposed not only a major challenge, but also a massive coordination effort, which was completed with the first proton-proton collisions reconstructed by the HLT. Furthermore, this thesis is completed with the study and implementation of on-line high transverse momentum triggers.

Zusammenfassung

Ein existenzieller Wesenszug des Menschen ist die Neugier auf das Unbekannte und das Neue. Deshalb versuchen von je her von Forscher und Wissenschaftler die Rätsel des Universums zu ergründen und die Welt, so wie wir sie heute kennen, zu erklären. Wir glauben heute ein ziemlich umfangreiches Verständis darüber zu besitzen, wie sich die Erde und das Universum in den letzten 14 Milliarden Jahren entwickelt haben. Jedoch gibt es immer noch große weiße Flecken auf der Landkarte der Forschung und Wissenschaftler aller Disziplinen versuchen diese zu erforschen.

Zwei große Fragen, die sich Physiker stellen befinden sich an den entgegengesetzten Enden der Beobachtungsskala: Was ist das Innerste der Materie? - und Wie hat sich das Universum nach dem Urknall entwickelt? Beide werden parallel in der Hochenergie-Teilchen- und Kernphysik mit der Hilfe von Teilchenbeschleunigern untersucht.

Der größte und energiereichste, und damit der stärkste Teilchenbeschleuniger, der je gebaut wurde, ist der *Large Hadron Collider* (LHC) am CERN in Genf (Schweiz). Mit dem Start des LHCs hat eine neue Ära in der Hochenergiephysik begonnen. Vier Mehrzweckgroßexperimente sind in der Lage, Proton-Proton-Kollisionen bei nie gemessenen Schwerpunktenergien bis zu 14 TeV zu untersuchen. Damit ermöglicht er nicht nur die Suche nach dem Higgs Boson, sondern auch die nach neuer Physik außerhalb des bekannten Standard Modells der Teilchenphysik, sowie die Untersuchung der Struktur der Nukleonen.

Darüber hinaus ist der LHC auch in der Lage, Schwerionen bis hin zu Blei zu beschleunigen und miteinander kollidieren zu lassen. Dabei werden eine Energiedichte und Temperatur erzeugt, die Mikrosekunden nach dem Urknall, kurz nach dem elektroschwachen Phasenübergang, existierten. In diesem ausgedehnten Feuerball wird Materie transformiert und neue Materie erzeugt. Dieser Zustand von stark gekoppelter Materie wird von mehreren LHC Experimenten untersucht. Auf Grund der äußerst kurzen Lebenszeit dieses Materiezustands kann er nicht direkt beobachtet werden. Die Messung der Teilchen, die vom Medium emittiert wurden, erlaubt es, Rückschlüsse auf die Erzeugung und das Verhalten dieser extremen Form von Materie zu ziehen. In Kollisionen von Bleikernen werden am LHC die höchsten Energiedichten gemessen, die jemals in Schwerionenkollisionen erzeugt wurden.

A Large Ion Collider Experiment (ALICE), in dessen Kontext diese Arbeit entstand, ist eines der vier großen LHC Experimente. Es wurde entworfen und gebaut, um hauptsächlich Kollisionen von Schwerionen zu erforschen, ist aber auch in der Lage Proton-Proton (pp)-Kollisionen zu untersuchen. ALICE kombiniert verschiedene modernste Detektortechnologien in einem Experiment, um unterschiedliche Aspekte des in Schwerionenkollisionen erzeugten, stark gekoppelten, Mediums zu studieren. Um die enorme Anzahl von produzierten Teilchen zu bewältigen und zu unterscheiden, haben die einzelnen Detektoren und deren Datenauslese eine hohe Auflösung. Dies resultiert in Datenraten bis zu 26 GByte/s und die dadurch erzeugten Datenmengen bilden eine Herausforderung für die Datenverarbeitung und Datenanalyse.

Da solche Datenraten und Datenmengen nicht einfach gespeichert werden können, ist ein Auswahlmechanismus notwendig, vor allem auch deshalb, weil viele interessante

Prozesse relativ selten auftreten und sie von einer großen Anzahl an Hintergrundereignissen überdeckt werden. Aus diesem Grund wurde der ALICE *High-Level Trigger* (HLT) konzipiert und gebaut, um schon während der Datennahme die ausgelesenen Ereignisse online zu rekonstruieren oder komprimieren und die seltenen Prozesse zu selektieren, bevor die ausgelesenen Daten gespeichert werden. Indem damit die Anzahl der relevanten, gespeicherten Ereignisse erhöht wird, steigert sich die Leistungsfähigkeit und Ausbeute der einzelnen Datenanalysen.

Nach Jahren der Planung und Entwicklung startete die Inbetriebnahme von ALICE und ihren Sub-Systemen im Jahr 2006 und dauerte an bis 2010 - als die ersten Protonen mit einer Schwerpunktenergie von $\sqrt{s} = 7\,\mathrm{TeV}$ kollidierten. Der Hauptteil dieser Arbeit beschreibt die Inbetriebnahme des in den ALICE-Datenpfad eingebetteten HLTs. Ein weiterer Teil erläutert die Anwendung des HLTs als Trigger für Teilchen mit hohem Transversalimplus p_t.

Der HLT wurde als Höchstleistungsrechnerverbund, ein sogenannter Höchstleistungscluster, geplant und konstruiert. Die Inbetriebnahme des HLTs umfaßte somit eine weite Spanne von Aufgaben, die in Beziehung zu der Clusterinfrastruktur, insbesondere der Hard- und Software, stehen. Zusätzlich mußten auch das Daten-Transport-Framework und die Rekonstruktions- und Analyse-Frameworks in Betrieb genommen werden.

Da der HLT ein integraler Bestandteil des Datenpfads in ALICE ist, verfügt es über Schnittstellen, sogenannte Interfaces, zu fast jedem Sub-System, das an der Datenaufnahme beteiligt ist. Das korrekte Funktionieren dieser Schnittstellen ist äußerst wichtig um einen reibungsfreien Ablauf der Datenaufnahme zu garantieren. Eines der Schlüsselthemen der Inbetriebnahme war die Bereitstellung der Informationen von rekonstruierten Ereignissen, welche sowohl die rekonstruierten Ereignisse an sich als auch Histogramme zur Datenqualität beinhalten.

Tatsache ist, dass eine Inbetriebnahme eines solch komplexen Systems kein gradliniger Prozess ist, der aber eine klar vorgegebene Richtung benötigt. Während des Inbetriebnahmeverfahrens mußten die Entwurfsüberlegungen auf Grund neuer Entwicklungen im HLT auch im Gesamtsystem ALICE angepaßt werden. Zudem mussten neue Anwendungen entwickelt und gebaut werden um den gewonnenen Erfahrungen gerecht zu werden. Die Verwendung von *leading-edge*-Technologien, wie die neueste und innovativste Computerhardware, Netzwerk Technologie, sowie Grafikkarten (GPUs) fügte noch eine weitere Ebene der Komplexität hinzu. All dies führte zu einem herausfordernden Inbetriebnahmeverfahren, das fast täglich zu wichtigen Entscheidungen darüber führte, wie man weiter fortzusetzen habe.

Diese Arbeit motiviert nicht nur die Notwendigkeit für einen ALICE HLT, sondern beschreibt auch dessen erfolgreiche Entwicklung, Umsetzung und Inbetriebnahme. ALICE wurde konzipiert, um im ungünstigsten Fall eine Multiplizitätsdichte von geladenen Teilchen von $\mathrm{d}N_\mathrm{ch}/\mathrm{d}\eta = 8000$ bei einer Ausleserate von 200 Hz mit zentralen Pb–Pb-Kollisionen zu bewältigen. Im Falle von pp-Kollisionen betrug die geplante Ausleserate 1 kHz. Die Größe der einzelnen Ereignisse wird von der Datengröße des wichtigsten Spurdetektors der Zeitprojektionskammer (*Time Projection Chamber* (TPC)), dominiert. Die Ereignisgröße der TPC ist fast linear abhängig von der Anzahl der geladenen Teilchen im Driftvolumen. Mit einer langen Driftzeit von 94 μs ist sie darüber hinaus anfällig für jegliche Art von Pile-up, das heißt multiple Ereignisse innerhalb des Driftvolumens.

Im ungünstigsten Fall werden für zentrale Pb–Pb-Kollisionen bei $\mathrm{d}N_\mathrm{ch}/\mathrm{d}\eta = 8000$

Ereignisgrößen von $\approx 86\,\mathrm{MByte/Ereignis}$ erwartet. Im Durchschnitt haben sogenannte *minimum bias* -Ereignisse eine Multiplizität von geladenen Teilchen, die etwa 25% derer von zentralen Kollisionen entspricht. Dies führt zu einer durchschnittlichen Ereignisgröße von $\approx 22\,\mathrm{MByte/Ereignis}$, wenn man die lineare Multiplizitätsabhängigkeit der TPC und das Datenvolumen der anderen Detektoren berücksichtigt. Bei einer maximalen Luminosität von $\mathscr{L} = 10^{27}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ in Pb–Pb-Kollisionen wird eine Wechselwirkungsrate von 8 kHz erwartet. Die eigentliche Ausleserate ist dabei auf etwa 300 Hz für zentrale Kollisionen limitiert, was zu einer maximalen Datenrate von 26 GByte/s führt.

Die maximal erreichbare Bandbreite zur dauerhaften Speicherung der Daten beträgt $1,25\,\mathrm{GByte/s}$, von der angenommen wurde, dass sie ausreicht, um die notwendige Statistik für die verschiedenen Datenanalysen aufzunehmen. Gleichzeitig ist sie ein Kompromiss um Kosten und Leistung von den Massenspeicher-Systemen. Die Notwendigkeit zur Datenreduktion mit einem HLT manifestiert sich in der Differenz zwischen der maximalen Ausleserate und der maximalen Speicherbandbreite und kann durch Datenkompression oder Trigger-Algorithmen realisiert werden.

Um diese Aufgaben zu bewältigen, muss der HLT sowohl eine enorme Rechenkapazität als auch die Flexibilät haben, schnell zu reagieren, falls zum Beispiel die Trigger- oder Rekonstruktionsalgorithmen angepasst werden müssen. Weiterhin muss er höchst ausfallsicher und leicht erweiterbar sein. Auch deswegen wurde der HLT als Höchstleistungscluster konzipiert und gebaut, und mit mehreren Abstraktionsschichten ausgestattet, um zur Trennung von Hardware, Daten-Transport, sowie der Rekonstruktion- und Analysesoftware beizutragen. Um die benötigte Leistung zu erzielen, wurden vom HLT unterschiedliche Möglichkeiten der parallelen Datenverarbeitung auf der Daten-Transportebene sowie in der Rekonstruktionsoftware implementiert. Der HLT wurde vollständig in den ALICE-Datenpfad integriert und mit den anderen Online-Systemen durch ihre Schnittstellen verbunden.

Ein wichtiger Meilenstein wurde mit dem Abschluß der erfolgreichen Inbetriebnahme des ALICE HLT im Frühjahr 2010 durch die ersten pp-Kollisionen bei $\sqrt{s} = 7\,\mathrm{TeV}$ erreicht. Im Laufe des Jahres wurden Eingangsdatenraten in den HLT von bis zu $1,5\,\mathrm{kHz}$ und Ereignisgrößen bis zu $1,5\,\mathrm{MByte}$ in pp Kollisionen erreicht. Damit hat der HLT nicht nur die Designanforderungen erfüllt sondern sogar übertroffen.

Die wichtigsten Aspekte der erfolgreichen Inbetriebnahme des funktionierenden HLT-Prototyps im Zusammenhang mit dieser Arbeit sind der Aufbau und die Inbetriebnahme eines Höchstleistungsclusters, das Design und die Inbetriebnahme des Konfigurationssystems und RunControl des HLTs, das Design und die Implementierung von Verteilungsmechanismen für die Anwendungssoftware, die Integration der ALICE Detektoren und der Schnittstellen zu den anderen Online- und Offline-Systemen, die Bereitstellung von online rekonstruierten Ereignissen, sowie die Entwicklung der ersten Trigger-Anwendung in pp-Kollisionen.

Als moderner HLT ist auch der ALICE HLT als skalierbarer Höchstleistungscluster mit hoher Verfügbarkeit und trotzdem niedrigen Anschaffungs- und Betriebskosten konzipiert und gebaut worden. Er folgt der sogenannten *Beowulf*-Architektur, die einen Cluster aus handelsüblichen Rechnern beschreibt, welcher mit einem Standard-Netzwerk, einem Linux- oder Unix-basierten Betriebssystem sowie Open-Source Softwareanwendungen ausgerüstet und von einem einzigen Steuerknoten aus betrieben wird. Im Gegensatz zu der ursprünglichen Beowulf-Architektur ist die Software für die parallele Datenverarbeitung im HLT nicht MPI basiert. Statt dessen nutzt der HLT sein eigenes

Daten-Transport Framework, um die Verteilung der Ereignisfragmente auf die einzelnen Rechenknoten sicherzustellen.

Die HLT-Hardware besteht mehreren Komponenten. Die 121 sogenannten *Front-End Processor* (FEP) Knoten empfangen die ausgelesenen Detektordaten über 460 Glasfaserkabel und führen für die TPC die erste Stufe der Rekonstruktion in ihren FPGAs durch. 51 Rechenkonten mit ihren 408 Prozessorkernen führen die weitere Ereignisrekonstruktion durch und bieten Platz für die Trigger- und Kompressionsalgorithmen. Weiterhin stellen mehrere PCs die Cluster-Infrastruktur zur Verfügung, unter anderem die Benutzerverwaltung, die Netzwerkadressverwaltung sowie das verteilte Dateisystem AFS. Verbunden sind die einzelnen Knoten mit 3 verschieden Netzwerken: Einem Fast-Ethernet Netzwerk für Verwaltungs- und Instandhaltungsaufgaben, einem GigaBit-Ethernet Netzwerk für die Daten und einem auf InfiniBand basierten Backbone-Netzwerk um potentielle Engstellen im Daten Netzwerk zu vermeiden.

Während der normalen Datennahme werden Tausende von Prozessen benötigt, um die ausgelesenen Ereignisse auf mehreren hundert Knoten zu rekonstruieren. Diese Prozesse müssen synchron gestartet und gesteuert sowie laufend mit Statusinformationen und den Detektorbedingungen versorgt werden. Die Konfiguration dieser Prozesse und deren Zusammenspiel mit den Schnittstellen nach außen sind unter der *RunControl* zusammengefasst. Eine Hierarchie von sogenanten TaskManager-Prozessen steuert die einzelnen Datenverarbeitungsprozesse und deren Versorgung mit Informationen. Die oberste Einheit, der sogenannte *RunManager*, überwacht und steuert alle TaskManager und ist damit in der Lage den ganzen HLT zu steuern. Er kommuniziert über die *HLT-ECS Proxy* Schnittstelle direkt mit dem Steuerungssystem für den gesamten ALICE Detektor, dem *Experiment Control System* (ECS).

Die Konfiguration des HLTs für eine Datenaufnahme ist komplett durch das Triggermenü des HLTs bestimmt. Es legt fest, welche Trigger-Algorithmen zur HLT-Triggerentscheidung beitragen und deshalb auch welche Detektordaten, also Rekonstruktionsprozesse, benötigt werden. Bei dem Beginn der Datenaufnahme werden der HLT, und somit die TaskManager und die einzelnen Datenverarbeitungsprozesse jedes mal neu konfiguriert, um auf Änderungen der Detektor-Hardware reagieren zu können. Um dies effizient zu ermöglichen, wurde eine baumartige Struktur von modularen Konfigurationsobjekten angelegt, aus der die gesamte HLT-Konfiguration erzeugt werden kann. Die einzelnen Objekte enthalten Informationen zu den Datenverarbeitungsprozessen an sich sowie deren Abhängigkeiten untereinander.

Auf jedem der in der Datenaufnahme beteiligten Knoten muss eine konsistente Version der Anwendungssoftware vorliegen. Um dies sicherzustellen, wurden verschiedene Möglichkeiten untersucht und in Betracht gezogen. Sequentielle Kopiermechanismen und das verteilte Dateisystem NFS skalieren nicht mit einer großen Anzahl von Knoten. Baumartige Synchronisationsmechanismen, die Ausfälle von Verzweigungsknoten bewältigen und so immer jeden Knoten im Cluster erreichen, mussten zur damaligen Zeit erst noch entwickelt werden. Das verteilte Dateisystem AFS ist in der Lage, eine große Anzahl von kleinen und mittleren Dateien zu speichern. Sein System von beschreibbaren Datenvolumen und deren schreibgeschützten Kopien erlaubt einen lastverteiltedn Lesezugriff auf Letztere. Sein intrinsisches Design passt perfekt auf die Anforderungen des HLTs, da die Anwendungssoftware nur einmal installiert wird und dann von den Datenverarbeitungsprozessen lediglich gelesen wird.

Die Schnittstellen zu den anderen On- und Offline-Systemen spielen nicht nur eine

wichtige Rolle für die Datenverarbeitung selbst, sondern auch für das Konfigurationsverfahren. Sie können in drei Klassen nach den verwendeten Datentypen eingeteilt werden: Physikereignisse, Konfigurationsdaten und Detektorzustände. Im Laufe der Inbetriebnahme wurden alle FEP-Knoten zur Verfügung gestellt um alle ALICE-Detektoren mit dem HLT zu verbinden. Die Schnittstellen zu ECS, dem *Detector Control System* (DCS), dem FileExchangeServer sowie der *Off-line Conditions DataBase* (OCDB) liefern die benötigten Konfigurationsdaten und Detektorzustände für das Funktionieren des HLTs.

Ein wesentliches Ziel der HLT-Inbetriebnahme war die Bereitstellung von online rekonstruierten Ereignissen, was nicht nur die Detektoren während ihrer eigenen Inbetriebnahmephase unterstützte, sondern auch dem HLT selbst erlaubte, seine Leistung und Datenqualität zu studieren. Ferner können durch das Betreiben eines Online-Eventdisplays die Wechselwirkungen optisch in Echtzeit dargestellt werden.

Die Zugriffsmöglichkeiten sind in einen primären Datenpfad über die *Data AcQuisition* (DAQ) und einen sekundären Datenpfad, der die normale Datennahme nicht beeinflussen darf, über sogenannte *TCPDumpSubscribers* (TDSs) unterteilt. Die TDSs sind spezielle Prozesse, die in Onlinerekonstruktionshierarchie eingebaut werden und jede Art von Daten über TCP Ports zur Verfügung stellen können. Sie werden anhand ihrer Datenblöcke in synchrone (im Bezug zur EventID) und asynchrone TDSs kategorisiert. Erstere liefern alle Informationen, die zu einem einzigen Ereignis gehören, um es zum Beispiel in einem Eventdisplay darzustellen. Letztere ermöglichen es, Informationen, die über mehrere Ereignisse hinweg gewonnen wurden, zum Beispiel Histogramme zur Datenqualiät, zur Verfügung zu stellen.

Im Laufe der Datennahme 2010 steigerte der LHC seine Luminosität schneller als vorhergesehen und damit drohte sich der von ALICE für 2010 reservierte permanente Speicherplatz vor dem Ende der pp Datennahme zu füllen. Da in dieser Zeit keine große Anzahl von Hardware Triggern existierte, war die naheliegendste Möglichkeit um die Menge von gespeicherten Daten zu reduzieren, der HLT. Zu diesem Zweck wurde als Teil dieser Arbeit ein Trigger auf Teilchen mit hohem Transversalimpuls implementiert, getestet und in Betrieb genommen. Da die Inbetriebnahme eines jeden Beschleunigers ein nicht präzise vorhersagbarer Prozess ist, schritt der LHC am Ende nicht mit der gleichen Geschwindigkeit voran und es gab keine Notwendigkeit, die Menge der gespeicherten Daten zu verringern.

Es ist eine Herausforderung, ein solch großes und komplexes System zu testen und Benchmarks auszuführen, wenn kein Entwicklungssystem von ähnlicher Größe vorhanden ist. Der existierende Entwicklungscluster konnte nur eine eingeschränkte Rolle für grundlegende Leistungs- und Datennahmetests spielen. Um die Skalierbarkeit des HLTs in seiner vollen Größe zu testen, mußte der Produktionscluster verwendet werden. Mit diesem wurden weitergehende Tests und Benchmarks durchgeführt, da die meisten bei der Inbetriebnahme entstandenen Probleme oft nur in der größten Ausbaustufe und im Zusammenspiel mit den anderen Online-Systemen auftraten.

Weitere umfangreiche Benchmarks und systematische Studien mit dem Ziel, den HLT in die Lage zu versetzen, Schwerionenkollisionen verarbeiten zu können, lagen nicht im Rahmen dieser Arbeit. Sie wurden nach dem Ende des skizzierten Forschungsabschnitts durchgeführt und gingen wie geplant mit den Erweiterungen der Clusterhardware einher. Das Datennetzwerk wurde komplett auf InfiniBand aufgerüstet und die endgültige Anzahl der Rechenknoten wurde gekauft und installiert, darunter auch einige mit GPUs für den schnellen Online-TPC-Trackingalgorithmus.

In der Zwischenzeit läuft der HLT seit über zwei Jahren erfolgreich und hat auch die ersten beiden Schwerionen-Perioden mit Pb–Pb-Kollisionen bei $\sqrt{s_{NN}} = 2,76\,\text{TeV}$ pro Nukleon-Paar hinter sich gebracht. Weiterhin hat der HLT erfolgreich seine erste Aufgabe der Rohdatenkompression in der Pb–Pb Periode Ende des Jahres 2011 erledigt. Dies ist die erste wirkliche Anwendung im eigentlichen Sinne des HLTs nach der Beendigung der Inbetriebnahmephase, außer der Bereitstellung von Histogrammen zur Datenqualität. Seitdem werden mit Hilfe der Onlinerekonstruktion nur rekonstruierte Raumpunkte der TPC, sogenannte Cluster, anstelle von TPC Rohdaten gespeichert und damit ein Kompressionsfaktor des Datenvolumens von bis zu 4 erzielt.

Acknowledgements

Writing acknowledgements is a tough job to do. So many people have helped and supported me in my last years and I would like to name them all in the first line. However, due to technical constraints this is not possible and, therefore, just because some people appear later in this text, it does not mean that they deserve less gratitude.

"It was a very exciting time", that is what I hear a lot about the last years. We, that are thousands of physicists, engineers and technicians have been working together to build the LHC and its experiments. I'm very grateful to Volker Lindenstruth, who gave me the opportunity to take part in this adventure, and actively participate and steer the commissioning of the ALICE High-Level Trigger (HLT), which led to this work.

Being stationed at CERN for more than 3 years as Technical Coordinator of the ALICE HLT, basically living in the ACR and CR2, was an extraordinary but also very work intensive period of my life. I also want to thank Dieter Röhrich, my second mentor from the Norwegian part of the HLT project. Volker and Dieter, both actively supported me during this time with advice, knowledge, strength and motivation, as well as fighting spirit and restraint, when necessary. After the relocation of our group from Heidelberg to Frankfurt and from the physics department to the computer science department, I'm very thankful to Harald Appelshäuser, who accepted me to finish this work with him within the physics department.

I would also like to thank my parents Ute and Jürgen Thäder as well as my sister Kristina, who all have always supported me in many different ways over the last decade during my studies and especially during the time of this work.

Special gratitude goes to my girlfriend Sonja Beyer, who has stayed with me and supported me over the whole period of this thesis. Being several years at CERN and especially living 500 km, or even 1,000 km apart from each other, combined with unconventional and long working hours was a hard exercise for both of us. Thank you!

Everybody who got e-mails from me during the last years could find a quote below it. In the beginning of the commissioning the quote *"Success is dependent on effort."* of the ancient Greek poet Sophocles was the mission statement. I would like to thank the whole HLT crew from Heidelberg, Frankfurt, Bergen, Oslo, Calcutta, Capetown, Berkeley, Yale, Cagliari, and Frascati. Only the combined effort led us to the successful commissioning.

Timm Steinbeck and Matthias Richter, are both colleagues and friends, whom I owe more than a simple thank you. They have been always around to help, support and motivate me. Furthermore, they were also there to restrain and decelerate me when I went too far with my resolute attitude.

Later Sophocles was replaced, by Hawkeye Pierce : *"My kingdom for an intelligent octopus!"*. In the stressful period before the first collisions, much more tasks and

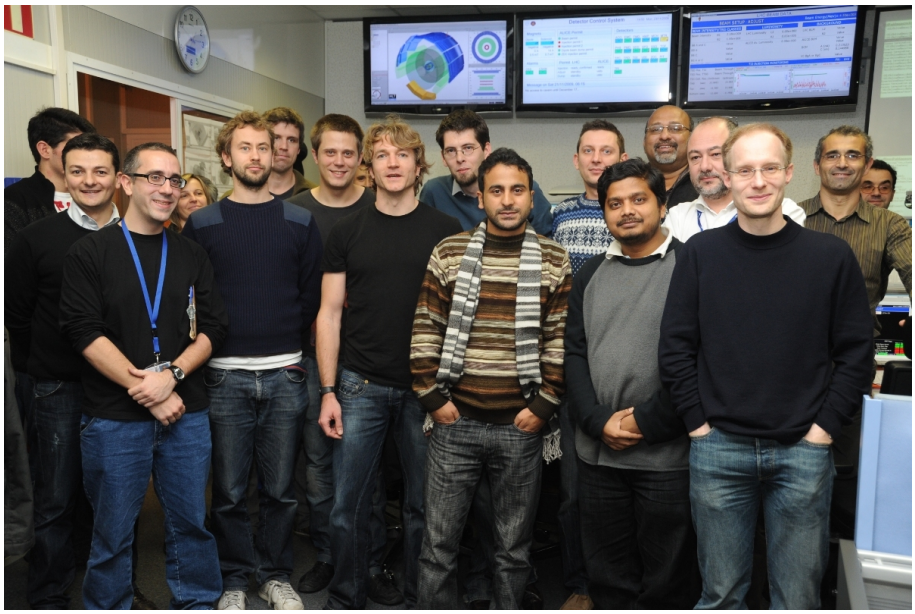

Figure 1: The HLT crew and more at the day of the first proton-proton collisions.

challenges appeared than foreseen. Here, I want to thank my friend and colleague Øystein Haaland, with whom I spent long working days and nights at P2. We were a good team, I think.

Thanks and respect also goes to Torsten Alt, who took the challenge after me to bring the HLT from a functioning prototype to a performing system.

All my friends, which were permanently at CERN like Stefan Kirsch, Magnus Mager, Christian Lippmann, Jens Steckert, Øystein Djuvsland, Jochen Klein, Per Thomas Hille and Paul Kuijer, being office mates, lunch mates or flat mates accounted for various support, help and friendship. Not to forget Kenneth Aamodt, Bruce Becker, Stefan Böttger, Timo Breitner, Indranil Das, Sergey Gorbunov, Kalliopi Kanaki, Sebastian Kalcher, Camilo Lara, Svein Lindal, Mamu, Arshad Masoodi, Gaute Øvrebekk, Mateusz Ploskon, Federico Ronchetti, Artur Szostak, Zeblon Vilakazi, and Pierre Zelnicek, as well as the whole P2 crew.

I'm very grateful to Matthias Richter, Timm Steinbeck, Ralf Averbeck and Jochen Klein, who spent a significant amount of time to read this thesis and to give valuable hints.

Last but not least, I want to thank Peter Braun-Munzinger and Silvia Masciocchi, who gave me the time and the necessary pressure to finish this work, together with the support of the whole GSI group.

Contents

1 Introduction **1**

2 Hadron and Heavy-Ion Physics at the LHC **3**
2.1 Hadron Physics . . . 3
2.1.1 Proton-Proton and Heavy-Ion Program at the LHC . . . 4
2.1.2 Hard Scattering Processes . . . 5
2.2 The Quark-Gluon Plasma . . . 5
2.2.1 Characteristics of the QGP . . . 6
2.2.2 Probes of the QGP . . . 7

3 A Large Ion Collider Experiment at the LHC **13**
3.1 Detector Layout . . . 13
3.1.1 Tracking Detectors . . . 15
3.1.2 Particle Identification . . . 16
3.1.3 Calorimetry . . . 17
3.1.4 Muon Spectrometer . . . 17
3.1.5 Multiplicity and Trigger Detectors . . . 17
3.2 On-Line Systems . . . 18
3.2.1 Data Flow Systems . . . 18
3.2.2 Control Systems . . . 21

4 The High-Level Trigger **23**
4.1 Architectural Design . . . 23
4.1.1 Design Goals . . . 23
4.1.2 Running Conditions . . . 23
4.1.3 Architectural Layers . . . 24
4.1.4 Hierarchical Structure . . . 24
4.2 Data Transport . . . 25
4.2.1 Data Input . . . 25
4.2.2 Data-Transport Framework . . . 26
4.2.3 Data Output . . . 27
4.3 Data Reconstruction and Analysis . . . 28
4.3.1 Massive Parallelization . . . 28
4.3.2 Reconstruction Strategies . . . 29
4.3.3 Detector Calibration . . . 30
4.3.4 On-line Analysis . . . 30
4.3.5 Full Event Reconstruction . . . 31
4.4 Trigger . . . 31
4.4.1 Trigger Framework . . . 32

4.4.2 Physics Triggers in the HLT 32

5 Commissioning of the HLT for pp collisions 35
5.1 Commissioning Concept 37
5.2 Commissioning Milestones 38
5.3 Hardware Integration 43
5.3.1 P2 - Counting Room Layout 45
5.3.2 Network 47
5.3.3 Nodes 49
5.4 Software Integration 53
5.4.1 Cluster Management 53
5.4.2 Application Software 55
5.5 Configuration 58
5.5.1 Interfaces 58
5.5.2 RunControl 62
5.6 Data Accessibility 73
5.6.1 Primary Data Path 73
5.6.2 Secondary Data Path 75

6 High-p_t Trigger 83
6.1 On-line Tracking Performance 83
6.1.1 Monte Carlo Simulations 84
6.2 Single Particle Trigger 86
6.2.1 Description 87
6.2.2 Results 89

7 Conclusions 93

A Trigger Naming 97
A.1 HLT Trigger Menu 97
A.2 HLT Trigger Class 97
A.3 HLT Trigger Menu for Cosmic Data-Taking Campaign 98

B Network Naming 101
B.1 IP Address Assignment Rules 101
B.2 ETHn Port Assignment Rules 102
B.3 IP Monitoring Proxy Port Assignement Rules 103

C Optical Fiber Connections 105

D Cluster Hardware Description 107
D.1 Cluster Nodes 107
D.1.1 Front-End Processors (FEPs) 107
D.1.2 Computing Nodes (CNs) 109
D.1.3 Infrastructure Nodes 109
D.2 Infrastructure Switches 111
D.3 Infrastructure External Connections 111

E Cluster Access **113**
E.1 User Access . . . 113
E.2 Operator Access . . . 114
E.3 On-line Data Access/Event Display . . . 115

F HLT Running States **117**
F.1 ECS States . . . 117
F.1.1 Detailed State Description . . . 117
F.2 TaskManager States . . . 120
F.2.1 Detailed State Description . . . 120

G RunControl Settings **123**

H HOMER Manager Implementation **129**
H.1 HOMER Source Descriptor . . . 129
H.2 HOMER Block Descriptor . . . 131
H.3 HOMER Proxy Handler . . . 134
H.4 HOMER Manager . . . 136

Glossary **141**

List of Figures **145**

List of Tables **147**

Bibliography **149**

Index **157**

1. Introduction

"I almost wish I hadn't gone down that rabbit-hole – and yet – and yet – it's rather curious, you know, this sort of life!", said Alice, a character in the fictional novel *Alice's Adventures in Wonderland* written by Lewis Carroll in 1865.

People always have been curious about the unknown and the new and hence it is no surprise that generations of scientists have tried to explore and explain the world as they discovered it. Today, we think to have a rather good understanding of how the earth and the universe have evolved over the last 14 billion years. However, there are still white spots and scientists from all disciplines try to explore those.

Two major questions asked by physicists are at the opposite ends of the observation horizon: What is the deepest inside of matter? How did the universe evolve after its creation at the Big Bang? Both are studied at the same time by high energy particle and nuclear physics with the help of large particle accelerators. The most powerful one ever built is the *Large Hadron Collider* (LHC) at CERN, Geneva, which reaches unprecedented energies allowing not only to search for the Higgs boson through proton-proton collisions, but also to study the inside of nucleons amongst other physics quests. Moreover, the LHC can also collide heavy-ions, creating an energy density and temperature, which existed microseconds after the Big Bang following the electro-weak phase transition. In this "fireball" existing matter is transformed and new matter is created. This forms a state of strongly coupled matter, which is investigated by several of the LHC experiments. However, it can not be observed directly due to its very short lifetime. The measurement of particles emitted from this medium allows to draw conclusions of the formation and behavior of this extreme matter.

One of the LHC experiments is the ALICE detector system, in which context the present work has been carried out. It combines state-of-the-art detector technologies in a dedicated heavy-ion collision experiment and allows to study many aspects of the produced strongly coupled medium. In order to cope with the large number of produced particles, ALICE has a high granularity read-out and, therefore, can deliver data rates up to 26 GByte/s. This enormous data volume represents a challenge for data handling and data analysis. As such data rates can not be easily stored, a selection mechanism is needed. Moreover, many of the interesting processes are rare and are hidden in background events. Therefore, the ALICE *High-Level Trigger* (HLT) has been designed to fully reconstruct the read out events and to be able to find those rare probes. The on-line reconstruction before storage allows to select and/or compress the relevant events and, therefore, to enhance the physics reach of ALICE by increasing the number of sampled events.

After years of planning and development, the commissioning of ALICE and its sub-systems started in 2006 and was carried out until 2010, when the first protons collided at a center-of-mass energy of $\sqrt{s} = 7\,\text{TeV}$. One part of this thesis describes

the commissioning of the HLT, which is embedded into the ALICE data path. Another part discusses the application of the HLT as trigger for particles with high transverse momentum p_{t}.

The HLT is designed as a high performance compute cluster and so the commissioning of the HLT included a large range of tasks related to the cluster infrastructure, i. e. hardware and software. In addition, data-transport, as well as the reconstruction and analysis software frameworks had to be commissioned. As the HLT is embedded in ALICE, it has interfaces to almost every subsystem in the data-taking process, which are very sensible for the functioning of the whole detector. One of the key issues of the commissioning process was the provision of access to the reconstructed event information to the collaboration.

As a matter of fact, the commissioning of a system with such a complexity is not a straightforward process, but needs to have a clear direction. In the process, design considerations had to be adjusted due to new developments, and new applications had to be built on the basis of new experience. Using leading-edge technologies, like latest computer hardware, network, and graphics cards, added yet another layer of complexity. All of this resulted in a challenging process, which very often lead to major decisions on how to continue on almost daily basis.

This thesis is organized as follows. An introduction in the physics of proton-proton and heavy-ion collisions is given in chapter 2, with a focus on high transverse momentum physics. The ALICE detector and the reasoning for the need for an HLT is laid out in chapter 3, followed by a description of the design and the architecture of the HLT in chapter 4. A detailed description of the HLT commissioning and the achieved goals and milestones is given in chapter 5. The actual architecture, the on-line configuration, and the access to the reconstructed data are the main content of this chapter. An overview over and the implementation of HLT triggers for high transverse momentum are discussed in chapter 6. The experience and results from the commissioning, running and triggering of the HLT are summarized in chapter 7, completed with an outlook on future perspectives.

This work covers only the commissioning and development of the ALICE HLT up to the first proton-proton collisions at $\sqrt{s} = 7\,\mathrm{TeV}$. However, in the meantime the HLT has been running successfully for over two years and also the first two run periods with heavy-ion collisions passed by with Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ per nucleon pair. The HLT successfully completed its first task of raw data compression in the Pb–Pb run end of 2011.

2. Hadron and Heavy-Ion Physics at the LHC

This is the era of the LHC, the *Large Hadron Collider* at CERN, with the ability of accelerating hadrons and nuclei to unprecedented energies, reaching a new energy regime. Hadron and heavy-nuclei collisions allow to test the *Standard Model* and to probe for physics beyond it.

The Standard Model [1] with its 19 free parameters (without neutrino masses) describes very successfully the present understanding of particle physics. It combines the electromagnetic and weak interactions in the electro-weak theory together with the strong force, described by *Quantum Chromodynamics* (QCD). Even if it agrees astonishingly well with the measurements, there are still open points, e. g. an experimental proof for the Higgs-Mechanism, which need to be further studied at the LHC, as well as potential physics beyond the Standard Model. Furthermore, heavy-ion collisions can be used to characterize partonic matter at extreme energy densities.

2.1 Hadron Physics

High energy particle physics is subject to a rhythm of lepton and hadron colliding experiments. Hadron accelerators like the SPS[1], the Tevatron[2], or the LHC are called "discovery machines". Colliding high energy hadrons (in general protons), the partons themselves (the quarks and gluons) interact, carrying a varying fraction of the energy of the hadron. Therefore, a wide energy range of parton-parton interactions is covered, leading to room for new discoveries. The hadron accelerators are followed in the cycle by lepton accelerators, like the *Large Electron Positron Collider* (LEP) at CERN, which collided electrons and positrons. As leptons are fundamental particles, they are point-like probes and the collision energy can be adjusted very well. Therefore, an energy range can be scanned for precise measurements. The field is completed by heavy-ion colliders like the RHIC[3] or the LHC, which are used to study nucleus-nucleus collisions.

A key issue in QCD is the introduction of the color charge (red, green, and blue, respectively anti-red, anti-green, and anti-blue) of the partons. However, quarks are never observed as isolated particles. They are confined in colorless bound states, the so-called hadrons, where a quark–anti-quark ($q\overline{q}$) state is called a meson and a combination of three quarks a baryon. The form of the potential V_s to keep the quarks bound is given by

$$V_s = -\frac{4}{3}\frac{\alpha_s}{r} + kr \quad ,$$

where α_s is the coupling constant of the strong interaction, k a constant, and r the distance of two partons [2]. For small r the first term dominates, similar to the Coulomb

[1] Super Proton Synchrotron at CERN, Geneva, Switzerland
[2] $p\overline{p}$ collider at FermiLab, Chicago, United States
[3] Relativistic Heavy Ion Collider at BNL, Brookhaven, United States

potential of the QED[4]. The quarks are then *quasi-free* inside hadrons, which is the so-called *asymptotic freedom*. Going to larger r, by trying to remove one quark from the hadron, the potential rises linearly with r. The energy stored in this color field at some point exceeds the threshold to produce a $q\bar{q}$ pair. Then the string will break into to smaller strings, creating a new $q\bar{q}$ pair. This is schematically shown in Figure 2.1.

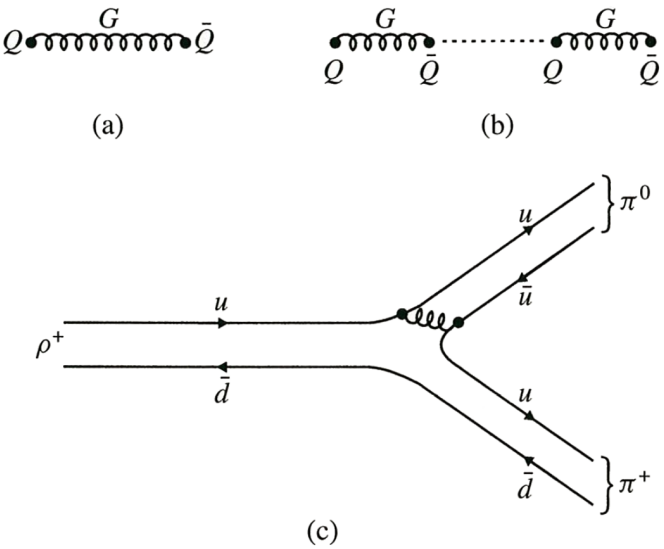

Figure 2.1: **a)** Two quarks connected by a gluon string, where one quark is moved away from the other. **b)** The string breaks and a new $q\bar{q}$ pair is created, as it is energetically favorable. **c)** This string breaking is illustrated by the decay of a ρ^+ ($u\bar{d}$) into π^+ π^0. Here a new $u\bar{u}$ pair is created via a gluon string. From [2].

2.1.1 Proton-Proton and Heavy-Ion Program at the LHC

Quantum Field Theory (QFT) is the framework of all particle physics theories, describing particles and fields in a consistent way [3]. In this theory, interactions between particles are mediated by gauge fields, or gauge particles. In the electro-weak theory these are the photon γ and the heavy gauge bosons W^+, W^-, and Z^0. These gauge fields Ψ are invariant under certain symmetry transformations, the so-called *local gauge transformations*:

$$\Psi \longrightarrow e^{i\theta(x)}\Psi \quad ,$$

where the phase factor θ (a real number) depends on the position in space-time x^μ. However, this is only true if the masses of the gauge bosons are zero, which is the case for the γ, but not for the W^+, the W^-, and the Z^0. Their masses are supposed to be created by the process of *spontaneous symmetry-breaking* of the *Higgs field*, a mechanism postulated already in 1964 [4]. The confirmation of the existence of the Higgs-boson mediating the Higgs field is one of the major goals of the LHC.

A further topic is the more detailed study of the top-quark, which was not long ago discovered at the Tevatron [5]. Its life-time is of the order of 10^{-25} s, which is shorter than the formation time of hadrons, such that no bound states involving top quarks

[4]Quantum Electrodynamics

can be created. The study of the *CP-violation* of the beauty quark system is another important measurement, helping to understand the matter – anti-matter asymmetry within the universe. Especially, as the LHC is a discovery machine, also physics beyond the Standard Model can manifest itself in the LHC experiments.

Most of the running time of the LHC is reserved for the *proton-proton* (pp) program, to study these open points. However, the study of the QCD also includes the observation of matter under extreme conditions as produced in heavy-ion collisions, which is carried out at the LHC as well. Here a new state of matter is investigated, the *Quark-Gluon Plasma* (QGP), which is discussed in detail in the next section. Nevertheless, measurements of pp collisions are an important reference to understand medium effects of the QGP.

2.1.2 Hard Scattering Processes

High energy hadron collisions can be classified by their scattering types. In elastic collisions, the initial- and final-state particles are of the same type. Inelastic collisions, where the hadrons are excited or even break up, can be divided into soft and hard collisions. Here the partons (quarks and gluons) interact directly. Interactions with a low momentum transfer Q are called *soft*, whereas collisions with large momentum transfer are classified as *hard interactions* [6, 7].

Theoretically, hard scatterings can be described with perturbative QCD (pQCD) calculations. The scattered partons in a hard interaction bear a high transverse momentum[5] p_{t} and the newly created particles emerge in opposite direction in the azimuth ϕ. They, together with the *initial-* and *final-state radiation*, form the hard part of the collision which can be evaluated with next-to-leading-order (NLO) pQCD calculations. The rest of the initial hadrons, after their break up, produce low momentum particles, which form the so-called *underlying event*. A schematic view of a hard collision is depicted in Figure 2.2.

All final-state partons undergo a non-pertubative hadronization process, forming colorless particles. Therefore, the outgoing high-p_{t} partons hadronize into sprays of particles, the so-called *jets* [8, 9]. Historically, this process is called fragmentation, which includes both final-state radiation and hadronization. Fragmentation functions describe the momentum distribution of hadrons within jets. The study of jet fragmentation functions in central heavy-ion collisions, compared to jets in pp collisions substantially contributes to the understanding of the quenching mechanism in the created medium. In this work the triggering on high-p_{t} particles is discussed with the goal to enhance the access to this kind of processes.

2.2 The Quark-Gluon Plasma

The concept of asymptotic freedom in the QCD has an interesting consequence. Above a large energy density at high temperatures or high baryon densities hadronic matter will dissolve in their constituents. This hot and dense medium of strongly coupled quarks

[5] The transverse momentum p_{t} of a particle with momentum $\overrightarrow{p} = (p_{\mathrm{x}}, p_{\mathrm{y}}, p_{\mathrm{z}})$ is defined as

$$p_{\mathrm{t}} = \sqrt{p_{\mathrm{x}}^2 + p_{\mathrm{y}}^2} \quad .$$

The beam axis is in z-direction.

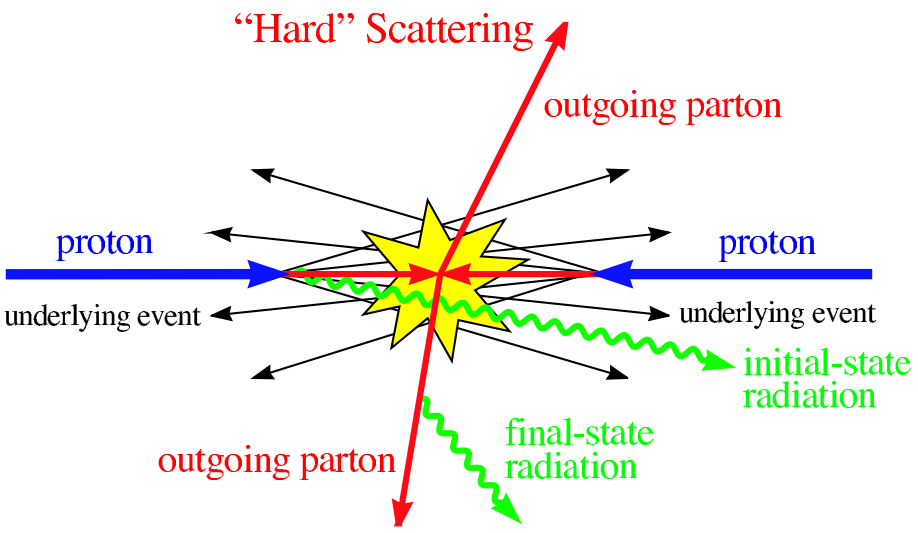

Figure 2.2: Sketch of a hard scattering event. From [6].

and gluons is commonly referred to as a *Quark-Gluon Plasma* (QGP) [10–12], similar to the electromagnetic plasma of free electrons and nuclei.

2.2.1 Characteristics of the QGP

It was proposed [13, 14], that when matter is brought up above a critical temperature $T_c \approx 170\,\text{MeV}$, it undergoes a phase transition and forms a strongly coupled medium with partons being the relevant degrees of freedom. In this medium the chiral symmetry should be restored and light quarks are expected to regain their very small current masses opposed to their large constituent masses, when bound in hadrons. This medium is the expected state of the early universe microseconds after the Big Bang, following the electro-weak phase transition. However nowadays, it can be only recreated in high energy heavy-ion collisions, as e^+e^- or pp collisions do not produce high enough energy densities over an extended volume.

After the formation of the fireball, made up of a large number of particles, it reaches a local equilibrium through multi-particle collisions and can therefore be described by thermodynamics. Initially, it was assumed that the created medium should behave like an ideal gas. However, results from RHIC [15] and also from ALICE [16] showed, that the measured characteristics can be described very well with hydrodynamical models using parameters for an almost ideal liquid. Therefore, one can conclude that the medium behaves more like an ideal fluid.

Shortly after the QGP is formed in heavy-ion collisions, it expands and cools down. The quarks and gluons get bound in mesons and baryons when the system crosses the critical temperature. This process is called hadronization, where in thermodynamics this change of state is referred to as phase transition. Figure 2.3 depicts the QCD phase diagram of strongly interacting matter with the relevant observables being net baryon density and temperature. Normal matter as we know it, is shown as a black dot at small temperatures. The area of the phase boundary between the hadronic phase and the medium is smeared, as the transition has not been observed so far. Further studies

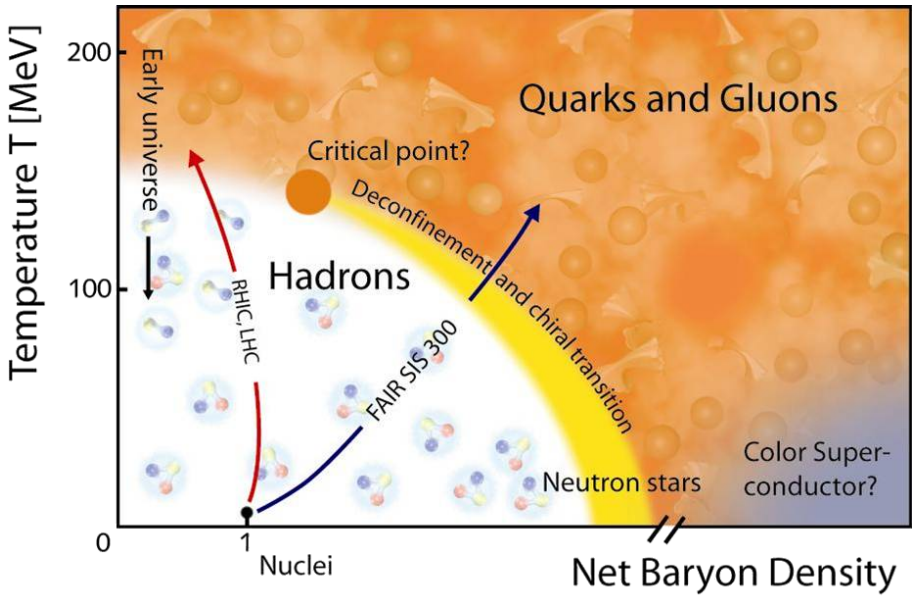

Figure 2.3: The schematic QCD phase diagram of strongly interacting matter. Normal nuclear matter is shown as a black dot at low temperatures. The transition area investigated at the LHC is indicated as red arrow. From [17].

of the medium created in heavy-ion collisions will allow for more insight in the phase diagram.

While the early universe cooled down from very hot temperatures at small baryon densities, today's high energy experiments use the high density and size of heavy nuclei to produce a measurable volume of the QGP. In future fixed target experiments at the FAIR[6] accelerator complex, a different region of the phase diagram is probed, creating a very dense medium at medium temperatures.

2.2.2 Probes of the QGP

The equilibrium state of the QGP lasts only for a very short time, of the order of 10^{-23} s and is therefore not directly accessible in heavy-ion experiments. Several probes of the QGP [18] are investigated in heavy-ion collisions. A selection is given here.

Particle Multiplicities

A basic global observable, is the average multiplicity of charged particles per unit rapidity[7], $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}y$. Its measurement allows to estimate the initial energy density ϵ in

[6] Facility for Antiproton and Ion Research at GSI, Darmstadt, Germany

[7] The rapidity y of a particle, with energy E and longitudinal momentum p_z relative to the beam axis is defined as

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right) \quad .$$

However, in experimental high energy particle physics the rapidity y is often replaced by the pseudo-rapidity η in the limit $p \gg m$:

$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right] = \frac{1}{2} \ln \left(\frac{|\vec{p}| + p_z}{|\vec{p}| - p_z} \right) \approx y \quad ,$$

where θ is the polar angle relative to the beam axis.

the overlapping, transverse area A of the created system in thermal equilibrium using Bjorken's formula [19]

$$\epsilon = \frac{\langle E_\mathrm{t} \rangle}{A \tau_0} \frac{\mathrm{d}N}{\mathrm{d}y}\bigg|_{y=0} \quad ,$$

where τ_0 is the formation time of the system and $(\mathrm{d}N_\mathrm{ch}/\mathrm{d}y)_{y=0}$ is the charged particle multiplicity at mid-rapidity with average transverse energy $\langle E_\mathrm{t} \rangle$.

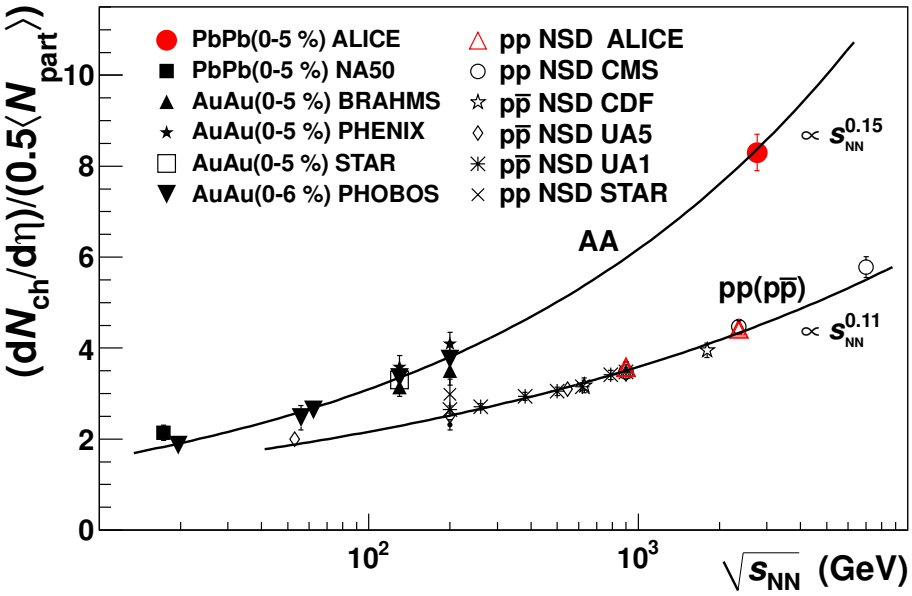

Figure 2.4: ALICE has measured the charged particle pseudo-rapidity density per participant pair at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ in central Pb–Pb collisions. The measurement extends the previous data from lower $\sqrt{s_{NN}}$ to LHC energies. From [20].

The charged particle pseudo-rapidity density per participant pair has been measured by ALICE at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ in central Pb–Pb collisions [20] and is shown in Figure 2.4. The result in Pb–Pb collisions at center-of-mass energy[8] per nucleon pair $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ is compared to previous measurements at SPS and RHIC, as well as to pp and p$\overline{\mathrm{p}}$ results from SPS, RHIC, Tevatron, and LHC. One can see a clear difference in the energy dependence of the charged particle multiplicity densities in pp

[8]The center-of-mass energy $\sqrt{s}$ in a collision of two particles with the four-momentum vectors $(E_1, \overrightarrow{p}_1)$ and $(E_2, \overrightarrow{p}_2)$ is defined as

$$\sqrt{s} = \sqrt{(E_1 + E_2)^2 - (\overrightarrow{p}_1 + \overrightarrow{p}_2)^2} \quad .$$

With both particles being from the same type $m = m_1 = m_2$ and in a collider with $E = E_1 = E_2$ and $\overrightarrow{p}_1 = -\overrightarrow{p}_2$ follows

$$\sqrt{s} = 2E \quad .$$

While accelerating heavy-ions with the atomic mass A and the atomic number Z, only the fraction Z/A of the energy of a proton beam with energy E_p is reached per nucleon pair (NN). The center-of-mass energy per nucleon pair $\sqrt{s_{NN}}$ is then

$$\sqrt{s_{NN}} = 2\frac{Z}{A}E_\mathrm{p} \quad .$$

and heavy-ion collisions, which points to the creation of a different system in heavy-ion collisions. The density of charged particles at mid-rapidity in central Pb–Pb collisions was measured by ALICE to be $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 1584 \pm 4\ (stat.) \pm 76\ (sys.)$ at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ [20].

Hydrodynamic Flow

In case of head-on collisions, the system is rotationally symmetric and the medium expands isotropically while cooling down. However, in the case of mid-central collisions[9], the nuclear overlap region has an almond shape and creates an anisotropic pressure gradient in the medium. This leads to an anisotropic expansion. These collective, so-called *flow* effects are well described by hydrodynamic calculations. The asymmetry of particle production in the transverse plane can be decomposed into Fourier components of the density distribution [22]

$$E\frac{\mathrm{d}^3N}{\mathrm{d}^3p} = \frac{1}{2\pi}\frac{\mathrm{d}^2N}{p_{\mathrm{t}}\mathrm{d}p_{\mathrm{t}}\mathrm{d}y}\left(1 + \sum_{n=1}^{\infty} 2\,v_n\,cos\left[n\left(\phi - \Psi_R\right)\right]\right) \quad ,$$

where E is the energy of the particle, p its momentum, p_{t} its transverse momentum, y is the rapidity, ϕ is the azimuthal angle, and Ψ_R is the azimuthal orientation of the reaction plane, which is defined by the direction of the impact parameter and the direction of the beam. The Fourier coefficients

$$v_n = \langle cos\left[n\left(\phi - \Psi_R\right)\right]\rangle$$

are dependent on p_{t} and y, and allow for an estimate of the viscosity of produced medium as well as the transverse flow velocity as a function of the emission angle relative to the reaction plane, $\phi - \Psi_R$. The first harmonic v_1 is called *directed flow* and is zero at mid-rapidity due to collision symmetry. The largest remaining coefficient is the second harmonic v_2, the *elliptic flow*, which describes the anisotropy in momentum space.

ALICE has measured the p_{t} integrated elliptic flow for 20-30% central Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ [16] and compared it to previous measurements at lower center-of-mass energies, as shown in Figure 2.5. It indicates a continued increase of v_2 from RHIC to LHC energies. Comparing the p_{t} dependence of v_2 with hydrodynamical models of the QGP, it turns out, that these describe the behavior of the observed strongly coupled medium rather well.

Suppression of Particle Production

The study of particle yields and, in particular, their kinematic distributions help to characterize the dynamics of a collision. The comparison of nucleus-nucleus (AA), proton-proton, as well as proton-nucleon (pA) collisions allows to disentangle between initial- and final-state mechanisms of particle production and dynamics [18, 23]. Hadrons with a high transverse momentum in general originate from hard scattering processes in

[9] Ultra relativistic nuclei are Lorentz contracted along the beam direction and are disk-like in the transverse direction [21]. Their radius can be approximated by $R \approx A^{1/3}$ fm. When two nuclei collide, the overlapping region is expressed in terms of the impact parameter b in the range of $[0, 2R]$ and as percentage of the nuclear inelastic cross-section. The collision is called most central if the nuclear overlap region is maximal ($b = 0$, 0% central) and ultra-peripheral if they just scrap ($b = 2R$, 100% central).

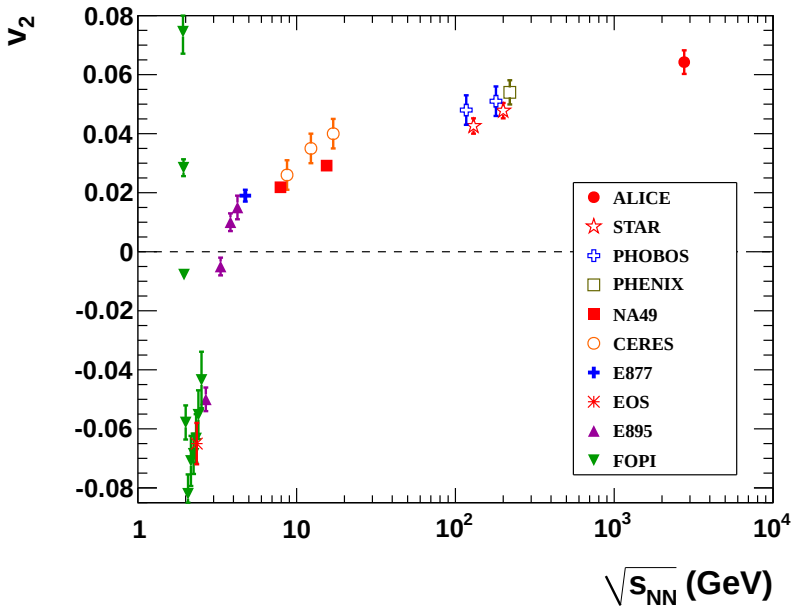

Figure 2.5: The p_{t} integrated elliptic flow has been measured by ALICE in Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ at 20-30% centrality. This measurement extends the data from previous lower $\sqrt{s_{NN}}$ to LHC energies. From [16].

the initial state of the collision. If an AA collision is different from just the superposition of independent nucleon-nucleon collisions, and a hot dense medium is formed in such a collision, initial-state particles are expected to loose energy while traversing this strongly coupled medium.

This comparison of yield ratios of high-p_{t} particles is quantified by the *nuclear modification factor* R_{AA}, which is defined as the ratio of the particle yield in nucleus-nucleus collisions and in pp collisions at the same energy per nucleon pair, normalized by $\langle N_{Coll} \rangle$ [26], the appropriate number of binary collisions:

$$R_{AA}(p_{\mathrm{t}}) = \frac{\mathrm{d}^2 N_{AA}/\mathrm{d}p_{\mathrm{t}}\mathrm{d}\eta}{\langle N_{Coll} \rangle \cdot \mathrm{d}^2 N_{pp}/\mathrm{d}p_{\mathrm{t}}\mathrm{d}\eta}$$

ALICE has measured the suppression of charged particles [24] in central Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$, as shown in Figure 2.6. A suppression of 0.14 is observed at $p_{\mathrm{t}} = 7\,\mathrm{GeV}/c$, which is much stronger than compared to RHIC results, interpreted as the evidence of a denser medium [27, 28]. The ALICE measurement extends significantly the transverse momentum reach and observes a rise of the R_{AA} for very high transverse momentum, which could not be observed conclusively at RHIC. For peripheral collisions, no significant suppression was measured. Measurements at RHIC from deuteron-gold collisions [29, 30], also show no significant suppression supporting the hypothesis of final-state modifications through the medium at RHIC.

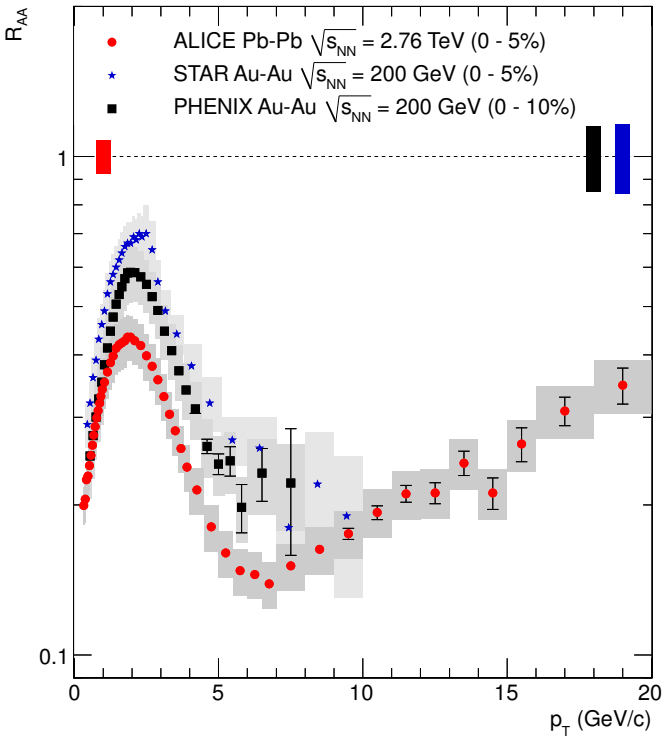

Figure 2.6: Suppression of charged particle production for high transverse momentum measured by ALICE. Compared to previous measurements from the RHIC experiments, a stronger suppression is observed. From [24].

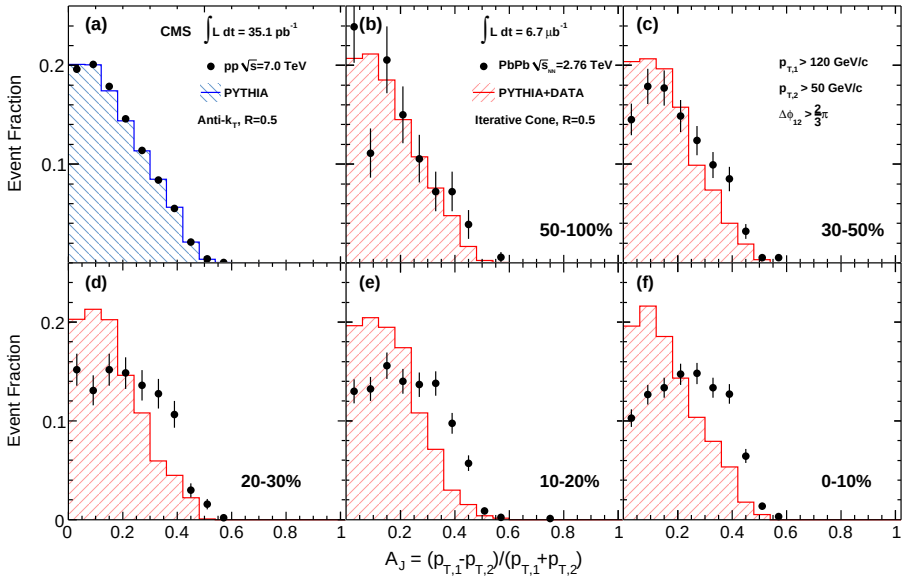

Figure 2.7: Di-jet asymmetry ratio measured by CMS for different centrality bins in Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$ for jets with $p_{\text{t},1} > 120\,\text{GeV}/c$ and $p_{\text{t},2} > 50\,\text{GeV}/c$ compared to pp collisions. A strong suppression for the most central bin is observed. Form [25].

Jet Suppression

In hard collisions of nucleons, normally two partons emerge in opposite direction in the azimuth ϕ. The fragmentation process leads to a spray of particles, the so-called jets. These parton jets traverse the strongly coupled medium and are both subject to energy loss due to interactions with it. Studying such di-jet events, the energy suppression can be quantified by the asymmetry ratio A_j

$$A_\mathrm{j} = \frac{p_{\mathrm{t},1} - p_{\mathrm{t},2}}{p_{\mathrm{t},1} + p_{\mathrm{t},2}} \quad ,$$

where $p_{\mathrm{t},1}$ is the transverse momentum of the jet with higher transverse momentum and $p_{\mathrm{t},2}$ of the one with the lower transverse momentum.

This observable for jet suppression was measured by CMS [25] and ATLAS [31] in Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$. The CMS results, requiring $p_{\mathrm{t},1} > 120\,\mathrm{GeV}/c$ and $p_{\mathrm{t},2} > 50\,\mathrm{GeV}/c$, are shown in Figure 2.7. No suppression is observed in pp collisions. Going from the most peripheral Pb–Pb collisions to the most central ones, an increasing asymmetry is observed.

3. A Large Ion Collider Experiment at the LHC

Four major experiments have been designed and built for the Large Hadron Collider (LHC) [32] (Figure 3.1) at the *European Organization for Nuclear Research* (CERN) in Geneva, Switzerland in order to explore a large variety of physics objectives.

LHCb[1] [33] is investigating the *CP*-violation in the *b-quark* system. ATLAS[2] [34] and CMS[3] [35] are multi-purpose detectors for high-energy physics, mainly concentrating their research on Standard Model and beyond Standard Model physics. The physics program of those detectors is focused on *proton-proton* (pp) collisions.

A Large Ion Collider Experiment (ALICE) [36–39] is as well a multi-purpose detector, mainly devoted to the physics of ultra relativistic heavy-ion collisions. However, ATLAS and CMS have an additional program to study heavy-ion collisions and ALICE also has an extensive program of measurements in pp collisions, which are also used as reference for heavy-ion collisions.

In the normal operation mode, the LHC is designed to accelerate bunches of protons, coming from the SPS accelerator with an injection energy of 450 GeV, up to 7 TeV per proton to reach 14 TeV center-of-mass energy. Once per year heavy-ions, i. e. lead ions, are accelerated from 177 GeV per nucleon up to 2.76 TeV per nucleon leading to a total center-of-mass energy of 1.15 PeV for fully stripped lead ions ($^{208}Pb^{82+}$) [32].

During the start-up phase of the LHC in 2009, the beams were not accelerated and collisions at the injection energy of 450 GeV per proton beam were established. Only in 2010, pp collisions with half the design energy, 3.5 TeV per proton beam, have been reached and have been the baseline energy until the end of the run period in 2011. Accordingly, the lead ion beams for the heavy-ion data-taking in November 2010 and 2011 had an energy of 1.38 TeV per nucleon, leading to a center-of-mass energy $\sqrt{s_{NN}}$ of 2.76 TeV per nucleon pair. The pp run period in 2012 has started with 4 TeV per proton beam.

3.1 Detector Layout

As ALICE is the only general purpose heavy-ion experiment at the LHC, it has to address a broad range of different physics observables and, therefore, implements a variety of different detector types with large dynamic range. Its total weight is approximately 10,000 t covering $16 \times 16 \times 26\,\mathrm{m}^3$.

ALICE was optimized for a charged particle multiplicity in central heavy-ion collisions at mid-rapidity with $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 4000$ but designed to cover also the range up to $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$. However, results from RHIC pointed to a significantly lower expectation of

[1] LHC beauty
[2] A Toroidal LHC ApparatuS
[3] Compact Muon Solenoid

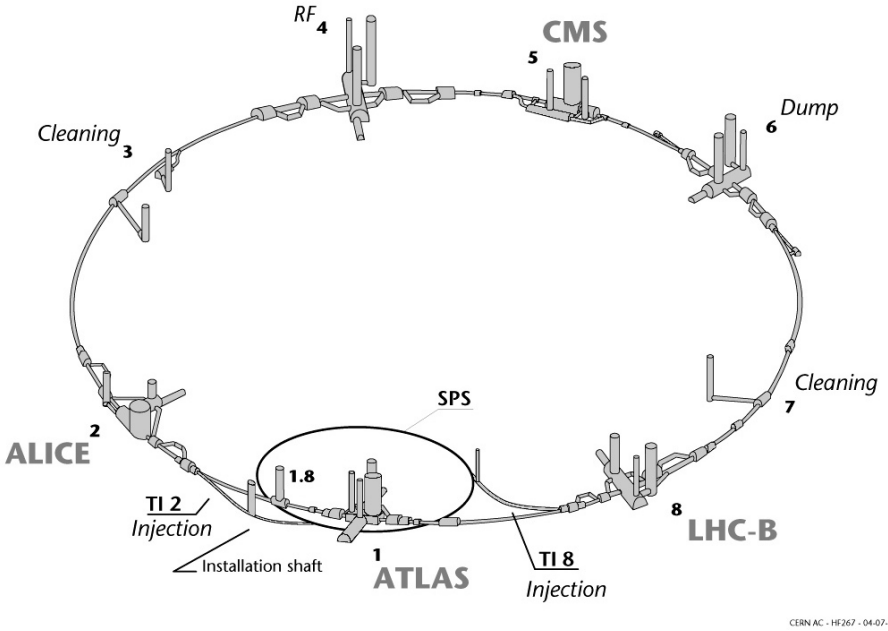

Figure 3.1: Overview of the LHC indicating the positions of the four large experiments CMS, LHCb, ATLAS, and ALICE at the *Interaction Points* (IPs) 5, 8, 1 and 2, respectively. In between the IPs 8 and 1 as well as between 1 and 2 are the transfer lines from the SPS (*TI 8* and *TI 2*). The beam cleaning areas are situated at the IPs 7 and 3. Acceleration takes place at IP 4 using radio frequency (RF) modules and the beam dumps are located at IP 6. Adapted from [40].

$\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 1500 - 4000$ [41, 42] for design energy, which is well in the dynamic range of ALICE. In the first Pb–Pb data-taking period, ALICE measured $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 1584 \pm 4(stat.) \pm 76(sys.)$ for half of the design energy [20](see section 2.2.2).

To fulfill the broad range of needs, the ALICE detector system (Figure 3.2) is divided into two parts, a central detector barrel and a forward muon spectrometer.

In the central part, tracking detectors, particle identification detectors, and electromagnetic calorimeters are situated inside the *L3 Magnet*, a solenoidal magnet with a moderate field strength of 0.5 T. Around the interaction point, in the middle of the L3 Magnet, the *Inner Tracking System* (ITS), the *Time Projection Chamber* (TPC), and the *Transition Radiation Detector* (TRD) form the ALICE tracking system covering the full azimuth. The TRD is mainly designed for electron identification and performs together with the TPC, the *Time-Of-Flight detector* (TOF) and the *High-Momentum Particle Identification Detector* (HMPID) the particle identification.

Two electromagnetic calorimeters, the *PHOton Spectrometer* (PHOS) and the *ElectroMagnetic CALorimeter* (EMCAL), complete the central barrel together with the multiplicity and trigger detectors and the *ALICE COsmic Ray DEtector* (ACORDE) mounted on top of the L3 Magnet. The muon spectrometer is situated in the backward direction towards the CMS detector.

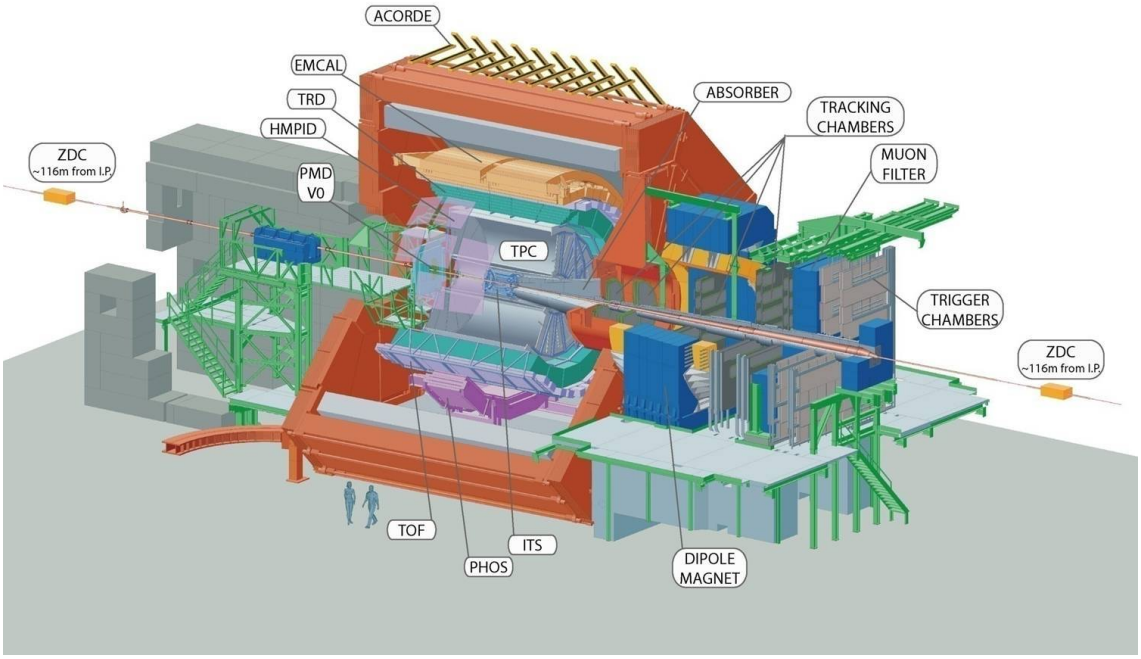

Figure 3.2: Schematic of the ALICE detector, showing the central barrel on the left side and the muon spectrometer on the right.

3.1.1 Tracking Detectors

The large multiplicity of charged particles, of the order of 10,000 in the acceptance of the central barrel, necessitates a robust tracking system. A TPC, which is a rather slow and data intensive device, together with high resolution silicon inner tracking detectors are well suited to provide the necessary capability of distinguishing two reconstructed tracks under such conditions. The additional use of the ITS and the TRD in the tracking process enhances the p_{t} resolution for high momentum particles. The total pseudo-rapidity coverage of the tracking system is $|\eta| < 0.9$.

Inner Tracking System Around the beryllium beam pipe, six layers of silicon detectors form the *Inner Tracking System* (ITS) covering the full azimuth [37, 43]. Its main tasks are the localization of the primary vertex with a resolution better then 100 μm, the reconstruction of secondary vertices from the decay of heavy flavor (D and B mesons) and strange particles, as well as the improvement of the momentum and angular resolution of the tracks found by the TPC. Being so close to the interaction point, the ITS is also capable of tracking and identifying low momentum particles below 200 MeV/c.

The ITS covers the pseudo-rapidity range $|\eta| < 0.9$ and is installed at radii between 3.9 cm and 42 cm from the interaction point. In order to cope with particle densities of 50 particles per cm^2 in the two innermost layers, these have been built as a *Silicon Pixel Detector* (SPD). Two layers of *Silicon Drift Detector* (SDD) and two layers of *Silicon Strip Detector* (SSD) comprise the four outer layers.

Time Projection Chamber The *Time Projection Chamber* (TPC) [37, 44, 45] is the main tracking device in ALICE covering the full azimuth and radii from $\approx 85\,\text{cm}$ to $\approx 250\,\text{cm}$, corresponding to a coverage in pseudo-rapidity of $|\eta| < 0.9$ for full radial track length in the momentum range of $100\,\text{MeV}/c$ up to $100\,\text{GeV}/c$. Additionally, the TPC delivers a $\mathrm{d}E/\mathrm{d}x$[4] resolution of the found tracks better than 5.5%, which allows for particle identification up to momenta of $20\,\text{GeV}/c$.

In order to reach the required momentum resolution the TPC is built out of a cylindrical Ne/CO_2 filled drift volume, which is divided by a high-voltage central electrode at 100 kV and equipped with a total of 557,568 read-out channels on the end-caps. The read-out is segmented into 216 partitions and comprises a total of 36 sectors (18 on each side), where 6 read-out partitions are grouped into one sector. Each channel is sampled in time, which is correlated with the z-direction parallel to the beam pipe, with 10 MHz to divide the long drift time of 94 μs in 940 samples.

The maximum trigger rate in pp collisions with an expected event size of $0.1 - 0.2\,\text{MByte}$ is limited to 3.5 kHz due to the space charge considerations. Event sizes up to $\approx 70\,\text{MByte}$ are expected in central Pb–Pb collisions. Already the enormous amount of read out data limits the maximum trigger rate to 300 Hz, resulting in a total of 21 GByte/s produced data. In 2010, average event sizes for minimum bias events of 580 kByte in pp and 11.5 MByte in Pb–Pb collisions have been measured with relaxed read-out settings [47, 48]. A larger background of beam-gas events for pp collisions in 2011 led to an average event size for minimum bias events of 1.6 MByte. An average event size for minimum bias Pb–Pb events of 13 MByte and 45 MByte for central events have been measured.

Transition Radiation Detector Surrounding the TPC, the *Transition Radiation Detector* (TRD) [37, 49] covers the full azimuth and the pseudo-rapidity range $|\eta| < 0.84$. Its main task is electron identification above $1\,\text{GeV}/c$ via the detection of transition radiation. The additional usage of secondary vertices, obtained with the ITS, allows the reconstruction of open-charm and open-beauty in semi-leptonic decays. Matching TPC, ITS refitted tracks to TRD tracks improves their position and momentum resolution. The very fast detector is also used to create Level 1 trigger decisions for high-p_t particles to enhance the yield of Υs, J/ψs, and jets in the recorded data set.

The TRD is segmented into 18 super-modules in azimuth following the TPC structure. Expected event sizes are 6 kByte in pp collisions and up to 11 MByte in central heavy-ion collisions. In 2010, only 7 super-modules were installed and were operated with relaxed read-out parameters. Average event sizes of 73 kByte in pp and 1.1 MByte in Pb–Pb collisions have been measured [47, 48]. In 2011, with 10 super-modules, average event sizes of minimum bias events of 110 kByte in pp and 1.5 MByte in Pb–Pb collisions have been determined.

3.1.2 Particle Identification

Besides the TPC, at a distance of $\approx 4\,\text{m}$ surrounding the TRD, the *Time-Of-Flight detector* (TOF) [37, 50, 51] performs the task of particle identification in the pseudo-rapidity region of $|\eta| < 0.9$ with full azimuthal acceptance. It is built of *Multi-gap Resistive Plate Chambers* (MRPCs) and has a time resolution of better than 100 ps.

[4] $\mathrm{d}E/\mathrm{d}x$ is the specific energy loss in a medium, described by the *Bethe-Bloch Formula* [46].

The main objectives of TOF are the identification and separation of pions, kaons, and protons in the intermediate momentum range (below $4\,\mathrm{GeV}/c$ for protons and below $2.5\,\mathrm{GeV}/c$ for kaons and pions).

Furthermore, the particle identification of charged hadrons can be improved by the *High-Momentum Particle Identification Detector* (HMPID) [37, 52] up to $5\,\mathrm{GeV}/c$ for protons and $3\,\mathrm{GeV}/c$ for kaons and pions. It is built as *Ring Imaging CHerenkov* (RICH) detector covering the azimuthal range $1.2° < \phi < 58.8°$ and the pseudo-rapidity range $|\eta| < 0.6$.

3.1.3 Calorimetry

Two electromagnetic calorimeters, both covering only parts of the acceptance, complete the detectors in the central barrel.

Photon Spectrometer In the lower region of the L3 Magnet, the *PHOton Spectrometer* (PHOS) [37, 53] covers the azimuthal range $220° < \phi < 320°$ and the pseudo-rapidity range $|\eta| < 0.12$ at a distance of 4.6 m from the interaction point. PHOS measures neutral mesons, thermal photons, as well photons from hard QCD processes. To cope with the high particle density at mid-rapidity PHOS implements a high granularity. It in its final form it consists of 17,920 $22 \times 22 \times 180\,\mathrm{mm}^3$ lead-tungstate ($PbWO_4$) crystals.

Electromagnetic Calorimeter Opposite in the azimuth of PHOS, the *ElectroMagnetic CALorimeter* (EMCAL) [37, 54] covers the the azimuthal range $80° < \phi < 187°$ and the pseudo-rapidity range $|\eta| < 0.7$. The Pb-scintillator sampling calorimeter has been designed to enhance ALICE's jet physics capabilities, measuring the neutral energy, together with the TPC, measuring the charged particles. With a lower granularity, but larger η acceptance than PHOS, it is optimized to study jet production rates and jet-quenching effects in heavy-ion collisions. Furthermore, it is also able to identify γ, π^0, η, and $\mathrm{e}^{\pm}$ particles.

3.1.4 Muon Spectrometer

The muon spectrometer [37, 55] is built in the backward direction towards the CMS detector covering the pseudo-rapidity range $-2.5 < \eta < -4.0$. It consists of a conical absorber reaching into the L3 Magnet, a large dipole magnet with a nominal field of 0.7 T just outside the solenoid, and five muon tracking stations before, in, and after the dipole. The muon arm is completed by two muon trigger stations behind an iron wall (the muon filter) used for muon identification and triggering. The muon spectrometer's primary goal is the measurement of the $\mu^+\mu^-$ decay channel of heavy-quark vector-meson resonances (i. e. J/ψ, Ψ', Υ, Υ', and Υ'') as well as semi-leptonic heavy flavor and the un-like sign dimuon continuum.

3.1.5 Multiplicity and Trigger Detectors

Several small detectors, specialized to measure global event characteristics and to trigger, complete the ALICE detector system. The *Forward Multiplicity Detector* (FMD) [37, 56] and the *Photon Multiplicity Detector* (PMD) [37, 57] measure the multiplicity of charged

particles and photons, respectively. The *T0* [37, 56] detector determines the event time and serves together with the *V0* [37, 56] detector as minimum bias trigger to discriminate against beam-gas interactions. A trigger on cosmic rays for calibration as well as cosmic ray physics is provided by the *ALICE COsmic Ray DEtector* (ACORDE) [37] on top the L3 Magnet. Inside the LHC tunnel, on both sides of ALICE at a distance of $\approx 116\,\mathrm{m}$ from the interaction point, the *Zero Degree Calorimeters* (ZDC) [37, 58] contribute to the centrality measurement in heavy-ion collisions and help to discriminate hadronic from electromagnetic interactions.

3.2 On-Line Systems

Five on-line systems support and allow the ALICE detectors to record physics interaction data and enable them to monitor and configure their detector hardware. They can be divided in the data flow systems and the control systems.

3.2.1 Data Flow Systems

The diversity of the different detectors systems, established in read-out time as well as data size, and also the different physics channels to be studied, account for ALICE's three-stage data-taking strategy. It allows for optimal data-taking rates for different physics processes even in central heavy-ion collisions. The general schema of the data flow can be seen in Figure 3.3.

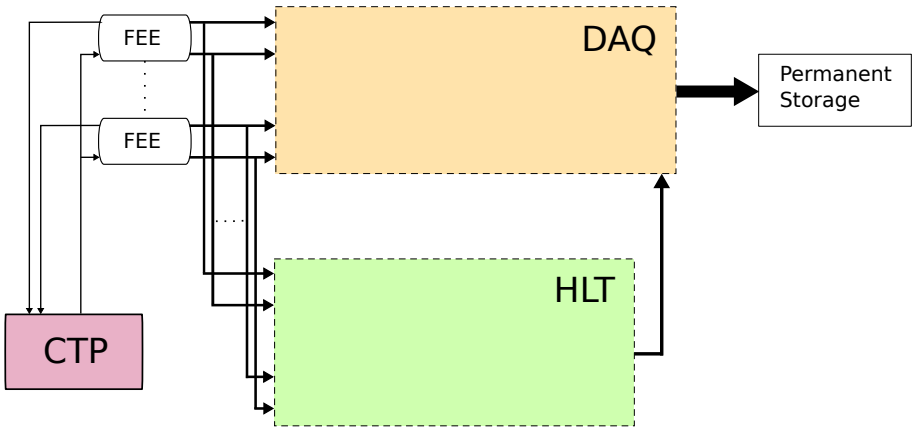

Figure 3.3: The *Front-End Electronics* (FEE) is triggered by the *Central Trigger Processor* (CTP) and sends the read out event fragments to the *Data AcQuisition* (DAQ), which immediately forwards a direct copy to the *High-Level Trigger* (HLT). Here a trigger decision is computed or data is compressed and sent to the DAQ, which then discards the event or starts the event building process. Finally, all accepted events are saved on permanent mass storage systems ready for off-line reconstruction and data-analysis.

Event Rates / Data Volume

Event rates and the data volume produced in ALICE vary significantly between pp and heavy-ion collisions. The event size of the TPC as main contributer to the data volume (more than 80%) changes strongly depending on the charged particle multiplicity.

pp In pp collisions the data volume depends strongly on the luminosity due to event pile-up in the TPC. At a luminosity of $\mathscr{L} = 3 \times 10^{30}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ an interaction rate of 140 kHz is expected, which leads to a pile-up of about 25 events in the TPC and, therefore, to an average event size of about 2.5 MByte/event. The event rate is limited to 1 kHz by the read-out rate of the SDD detector. The TPC itself is limited to a read-out rate of 3.5 kHz due to space charge considerations.

At the beginning of the 2010 pp data-taking at $\sqrt{s} = 7\,\mathrm{TeV}$, the average event size was measured to be 890 kByte/event at an average read-out rate of 845 Hz [47, 48]. The instantaneous peak luminosity was $\mathscr{L} \approx 10^{29}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ at an interaction rate of $\approx 6\,\mathrm{kHz}$ [59]. In 2011, the instantaneous peak luminosity increased up to $\mathscr{L} \approx 10^{30}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ [60].

Pb–Pb For central Pb–Pb collisions and a worst case assumption for the charged particle multiplicity of $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$, sizes of ≈ 86 MByte/event have been expected for the barrel detectors. On average, minimum bias heavy-ion collisions are expected to have a charged particle multiplicity of 25% of central collisions. As the TPC event size scales almost linearly with the number of charged particles, ALICE's average minimum bias event size is anticipated to be about 22 MByte/event. At a maximum luminosity of $\mathscr{L} = 10^{27}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ for Pb–Pb collisions an interaction rate of 8 kHz is expected. The actual read-out rate is reduced to about 300 Hz for central collisions which is limited by the TPC and 1 kHz for minimum bias events, limited by the read-out time of the TPC and other detectors.

In the first Pb–Pb run in November 2010 a peak luminosity of $\mathscr{L} \approx 2.5 \times 10^{25}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and an interaction rate of $\approx 200\,\mathrm{Hz}$ was reached [61]. The average minimum bias event size was measured to be 13.6 MByte/event at an average read-out rate of 115 Hz [47, 48]. In the second Pb–Pb run in 2011 the peak luminosity increased up to $\mathscr{L} \approx 4 \times 10^{26}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ with an interaction rate of $\approx 3.5\,\mathrm{kHz}$ [62].

These observations and assumptions lead to three extreme scenarios of read-out data volume:

- 1 kHz minimum bias pp collisions with about 2.1 GByte/s.
- 1 kHz minimum bias Pb–Pb collisions with about 22 GByte/s.
- 300 Hz central Pb–Pb collisions with about 26 GByte/s.

ALICE's maximum sustained bandwidth to the permanent storage is 1.25 GByte/s, which is estimated to be sufficient to provide the necessary statistics for the various physics channels and, at the same time, is a compromise on cost and performance for the mass storage systems. This emphasizes the importance of data reduction by the *High-Level Trigger* (HLT) to meet these requirements.

3.2.1.1 Trigger System

The *Central Trigger Processor* (CTP) [63] collects trigger inputs from the fast trigger detectors, computes a decision, and sends the read-out signal to all ALICE detectors. An anticipated interaction rate of 8 kHz ($\mathscr{L} = 10^{27}\mathrm{cm}^{-2}\mathrm{s}^{-1}$) in heavy-ion collisions and a large span of detector read-out times lead to a three-level design, where the Level 0

(L0) signal reaches the detectors after 1.2 μs. The Level 1 (L1) allows more time for more advanced trigger inputs after 6.5 μs.

Pile-up of central heavy-ion collisions in the TPC cannot be easily reconstructed due to the very high particle multiplicity. This can be taken into account by the *past-future protection* at the final trigger stage, the Level 2 (L2). After the drift time of the TPC of 88 μs the L2 *accept* (L2a) or L2 *reject* (L2r) signals are sent to the detectors.

3.2.1.2 Data Acquisition System

On the L2a signal, the detector read-out starts and the detector *Front-End-Electronics* (FEE) sends the detector data via 460 optical fibers to the *Data AcQuisition* (DAQ) [63] using a standardized ALICE protocol, the *Detector Data Link* (DDL) [64, 65]. The DAQ is structured in three layers, according to data locality. In the first level, the read out detector data is received by PCI-X[5] cards, the *DAQ Read-Out Receiver Cards* (D-RORCs), inserted in commodity PCs[6], called *Local Data Concentrators* (LDCs).

The LDCs build sub-event fragments and send them to the second level, the global event building, which is implemented as an event-building network of commodity PCs, the *Global Data Concentrators* (GDCs). In the final stage, the complete events are transfered via a fiber-channel storage network to permanent storage for later off-line reconstruction and analysis of the events.

In parallel, an exact copy of the incoming detector data in the D-RORC is sent via a second DDL on the same D-RORC to the HLT, which computes an even more sophisticated trigger decision and sends the decision and additional data to extra assigned LDCs. If an event gets rejected by the HLT, it is already discarded on the LDC level to spare the event building step. Three different run modes of the DAQ can be distinguished, determined by the activity of the HLT.

- DAQ only - HLT disabled (**Mode A**)
  No data is sent to the HLT and no HLT decision is expected by the DAQ.
- DAQ + HLT analysis (**Mode B**)
  Data is sent to the HLT, which processes the data, and computes a decision. However, the DAQ only checks for data integrity but does not consider the HLT decision.
- DAQ + HLT enabled (**Mode C**)
  Data is sent to the HLT, which processes the data, computes a decision, and the DAQ executes the decision.

3.2.1.3 High-Level Trigger System

The *High-Level Trigger* (HLT) [63] is a large high performance computing farm based on commodity PCs, situated in parallel to the DAQ in the data flow. It performs full reconstruction of the detector data on-line and allows for complex trigger and compression algorithms based on the full event information obtained from the reconstruction. Three main tasks of the HLT can be highlighted:

[5] Peripheral Component Interconnect-Extended
[6] Personal Computers

Filter Accepting or rejecting of events on the basis of the on-line reconstruction and analysis.

Select Selecting a physics *Region-Of-Interest* (ROI) within one event and, therefore, storing only part of the read out detector data.

Compress Reducing the event size with lossless and lossy compression algorithms applied on detector and reconstructed data.

The design considerations of the HLT and its system architecture are discussed in more detail in chapter 4.

3.2.2 Control Systems

Two control systems watch over the ALICE detectors and the other on-line systems as indicated in Figure 3.4.

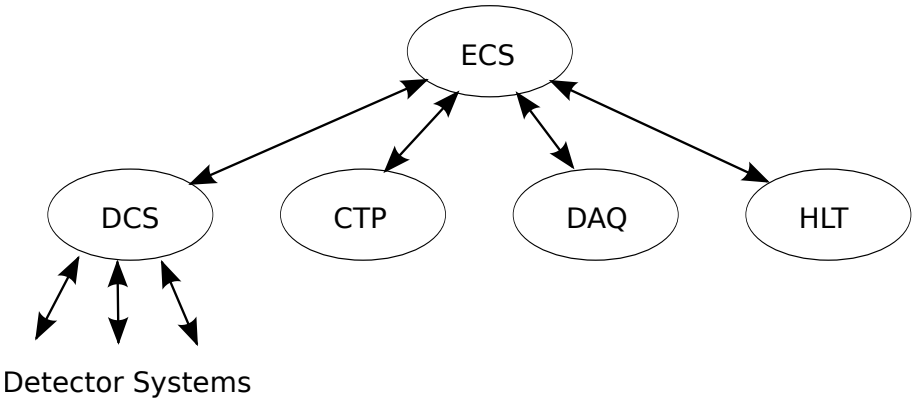

Figure 3.4: The *Experiment Control System* (ECS) is the top level entity steering the ALICE data-taking process. It controls and configures CTP, DAQ, and HLT directly and the detector systems indirectly via the *Detector Control System* (DCS).

Detector Control System The *Detector Control System* (DCS) [63] is responsible for the configuration and monitoring of the different detectors. Several parameters like temperature, pressure, and voltage of the detectors' subsystems are monitored and archived for the use in the on-line and off-line event reconstruction. Furthermore, the DCS handles ALICE's communication with the LHC.

Experiment Control System On top of all the detectors and on-line systems sits the *Experiment Control System* (ECS) [63] and naturally implements interfaces to all of them. This top control layer steers the data-taking and all other systems. It allows the operators in the *ALICE Control Room* (ACR) to configure the whole experiment via the *ALICE Configuration Tool* (ACT) and, therefore, to start and stop the data-taking process. The CTP, DAQ, HLT, and DCS are directly configured and controlled from the ECS, where the detectors are managed indirectly via the DCS.

4. The High-Level Trigger

This chapter describes the design and architecture of the ALICE *High-Level Trigger* (HLT) and introduces its main functional parts.

4.1 Architectural Design

The ALICE detector system will produce up to 26 GByte/s for central heavy-ion collisions and up to 2.1 GByte/s for minimum bias pp collisions, as summarized in section 3.2.1. In order to meet the maximum bandwidth to permanent storage of 1.25 GByte/s another level of event filtering is required which can execute complex trigger algorithms in contrast to the lower, hardware based trigger levels. Moreover, there is a need to execute sophisticated data compression algorithms on the read out data. This additional stage is implemented by the *High-Level Trigger* (HLT), which is described here. Further information can be found in [37, 63].

4.1.1 Design Goals

The reduction of the data volume by the means of selecting relevant data samples, e. g. low cross-section processes, is not a task which can be easily defined in a static way. In contrary, the requirements on the selection criteria will be evolving in a continuous process following the understanding of the recorded and analyzed data. Furthermore, the ALICE detector system currently consists of 18 sub-detectors (see section 3.1), each producing different amounts of data and at different rates.

In order to cope with these needs the HLT is designed as an open, flexible, scalable system, which minimizes the effort for future expansion, i. e. changes in the selection criteria up to ALICE detector upgrades. As ALICE's main data producer, the *Time Projection Chamber* (TPC) is in the main focus of HLT's effort. Nevertheless, almost all other sub-detectors are included in the HLT as well. To allow for complex trigger algorithms with changing criteria, the HLT has been designed as high-performance commodity PC farm. It performs a full reconstruction of all events on-line and applies trigger and compression algorithms on an event-by-event basis. The HLT utilizes current pattern recognition and event reconstruction techniques as well as data analysis and trigger algorithms in a modular way. Therefore, modules can be altered separately, new modules can be added and even interchanged in a simple way.

4.1.2 Running Conditions

The HLT allows ALICE to exploit the full luminosity at the maximum L2a trigger rate and then reduces this data to a storable amount. This data reduction is achieved by several means of event filtering (triggers and data compression), which define the different running scenarios of the HLT.

After the reconstruction of the event data, trigger algorithms are applied and allow to accept or reject full events. A finer grained decision is achieved by selecting only a *Region-Of-Interest* (ROI) of an event, i. e. selecting only some of *Detector Data Links* (DDLs) of an event, for permanent storage. Using lossless and lossy compression algorithms on raw[1] or reconstructed data also leads to a reduction of the data size, if the read-out, uncompressed data is discarded and only the compressed data is stored. The combination of the last two together with a detailed event information results in ROI inside the events DDL data, only selecting parts of the raw data of one event. This can be used to extract one piled up event in the TPC out of up to 25 events (see section 3.2.1).

For all this mechanisms the ALICE *Data AcQuisition* (DAQ) has to be in **Mode C**, which allows to fully exploit the HLTs capabilities. However, these filter algorithms have to be carefully tested which is done in **Mode B** (see section 3.2.1.2). Then no HLT trigger is applied, but the decision and additional HLT data is recorded for later analysis.

4.1.3 Architectural Layers

Following the modular design, the different reconstruction and trigger algorithms are distributed over the HLT computing farm. Three abstraction layers (see Figure 4.1) build the necessary infrastructure. The bottom layer is built from several hundred interconnected commodity PCs. On the top lies the *Reconstruction and Analysis Framework* which is responsible for the data processing itself. In between the two is a powerful, complex *Data-Transport Framework*, the core of the HLT. It is responsible for the efficient data handling and data transfer in between the computing nodes.

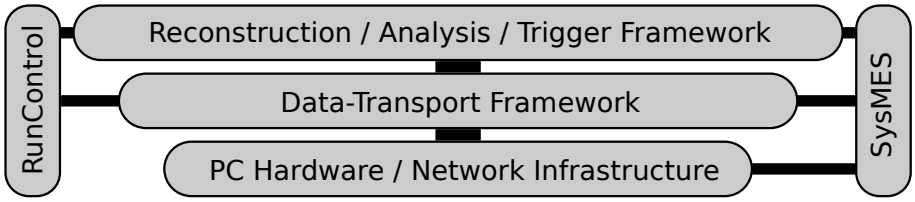

Figure 4.1: Overview of the three abstraction layers of the HLT, accompanied by the *RunControl* system and the cluster management system SysMES.

The data-taking and the communication with the *Experiment Control System* (ECS) is controlled by the *RunControl* which interacts with the *Reconstruction and Analysis Framework* and the *Data-Transport Framework*. All layers are monitored by the cluster monitoring and management system SysMES[2] [66].

4.1.4 Hierarchical Structure

The large amount of the detector data, which has to be reconstructed and analyzed, imposes a challenge to the HLT met by exploiting the concepts of parallelization, pipelining and modularity. Six independent stages implement a processing hierarchy in a

[1] data directly from the DDL
[2] System Management for Networked Embedded Systems and Clusters

treelike structure (outlined in Figure 4.2) as the event parts themselves are uncorrelated, in terms of data processing, in one stage as well as different stages. In the first stage the raw detector data is received according to the detector granularity. Using the principle of data locality, the first level, local reconstruction (stage 2), e. g. extraction of hit information or finding of clusters, is done directly after the arrival of the data. This is done in parallel for all incoming detector links. Already here the data size to be passed on to the next stage is reduced. This data reduction is continued in all further stages.

Every detector is reconstructed independently in the next stage followed by the global reconstruction stage. Here, all detector information is combined and an event summary object is created. Applying physics selection criteria on this, a trigger decision is computed in the fifth stage followed by a data compression stage.

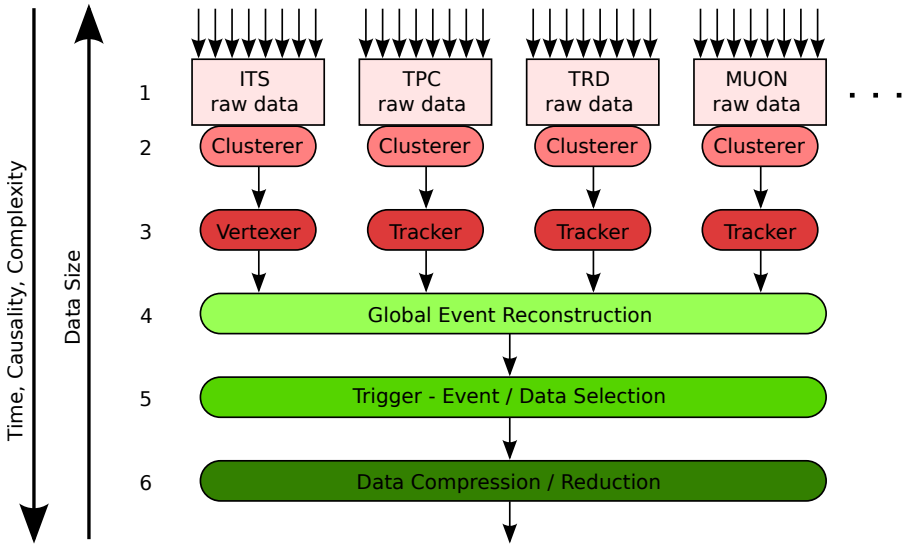

Figure 4.2: Overview of the HLT reconstruction hierarchy. Six stages implement a pipelined treelike structure, reducing the data size in every step.

In each stage, the different data processing tasks are structured in modular entities. These so-called *HLT components* are an integral part of the HLT. Each component has a well defined interface and is designed for a special task. They will be discussed in more detail in the next section.

4.2 Data Transport

As discussed in section 3.2.1.2, the HLT is embedded into the data path of ALICE. It is situated in parallel to the LDCs of the DAQ, but in sequence between the detector read-out and the event building.

4.2.1 Data Input

After an event is accepted by the L2a signal issued by the CTP, it is read out and sent from the *Front-End-Electronics* (FEE) of the detectors to the LDCs of the DAQ via optical fibers. This is done with a standardized protocol, the *Detector Data Link* (DDL) [64, 65], used in common for all ALICE detectors. The data is received by PCI-X cards,

the *DAQ Read-Out Receiver Cards* (D-RORC), which sent an exact copy of the data to the HLT before they store it in the LDCs.

In the HLT, the data is received by its input devices the *HLT Read-Out Receiver Cards* (H-RORC) [67]. These Xilinx Virtex-4 FPGA[3] based PCI-X cards are equipped either with two receiving (DIU[4]) or two sending interfaces (SIU[5]) each. Furthermore, they can also be used for data pre-processing, i. e. unpacking of raw data or even first level reconstruction. The H-RORCs are inserted into so-called *Front-End Processor* (FEP) nodes and write the received data directly into their main memory.

4.2.2 Data-Transport Framework

The communication, data transport, and load distribution between different HLT components is handled by the central layer of the HLT, the *HLT Data-Transport Framework* [68–70]. It is based on the Publisher/Subscriber principle where every process can announce (publish) data objects and another process can subscribe to it. This then creates the pipelined, treelike reconstruction chain of the components in the HLT. The components themselves are grouped in four types.

Data Source Components are the first components in the chain. They control the input from the H-RORC or load events from file into the main memory of the FEP nodes.

Data Processing Components are the main worker components. They subscribe to some output of preceding components, process the data, and publish the result to the next stage in the pipeline. The *AliRootWrapperSubscriber* component, as a part of the data-transport framework, defines a generic interface between the data-transport framework and the *HLT Components* of the *HLT Analysis Framework*, described in section 4.3. Following the modular structure of the HLT, all reconstruction and analysis processes are implemented such that the HLT components are independent of the data transport layer.

Data Flow Components are the heart of the data-transport framework. They control the routing of the data blocks through the reconstruction chain. Therefore, they handle the inter-node communication as well as the fan-outs and and fan-ins within the data flow.

Data Sink Components are the last stage in the processing. They handle the output of the chain and provide interfaces to the H-RORC as well as to generic TCP[6] ports.

In order to assure a low processing overhead of the data transport itself and to avoid unnecessary copy steps of output and input data on the same node, all data is kept in a shared memory [71]. As shown on the left side of Figure 4.3, a publisher writes its output directly in the shared memory and a subscriber can read from it. The descriptors of these data blocks are send via named pipes [71] between the different components to

[3] Field Programmable Gate Arrays
[4] Destination Interface Unit
[5] Source Interface Unit
[6] Transmission Control Protocol, a widely used network protocol

reduce the communication overhead. All data blocks and their descriptors are kept in the shared memory until the event was successfully processed by the HLT and transmitted to the DAQ to assure no loss of data in case of e. g. hardware failure, as re-routing of data blocks is foreseen.

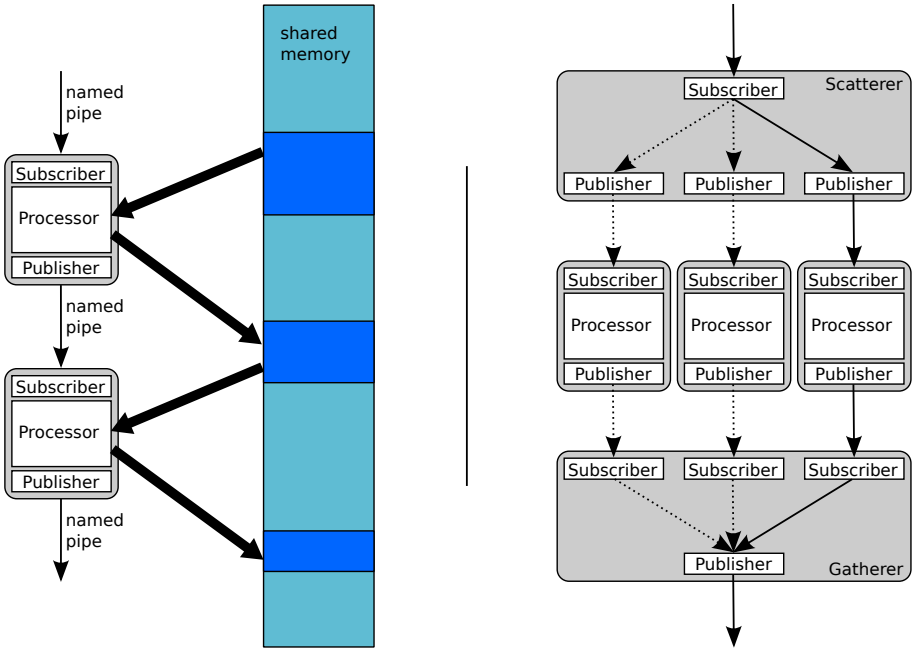

Figure 4.3: **Left side:** Data blocks between components are exchanged via shared memory and the associated descriptors via named pipes.
Right side: Scatterer components are able to fan-out the data path for load balancing. After possible intermediate processing steps the data stream is recombined by gatherer components. The solid line shows the path of one event.

In the inter-node communication a copying step is unavoidable which is a handled by specialized *Bridge Components*. The data is copied from the shared memory of one node into the shared memory of another node, such that the boundaries of the physical hardware are completely transparent for the processing components.

To allow for components slower than the event rate and for load balancing onto different nodes, the data path can be split as sketched on the right side of Figure 4.3. Special *Scatterer Components* distribute the incoming events to several outputs ports, according to a modulo on a unique event identifier, the EventID[7]. After intermediate processing and data flow components the data stream is merged again with the help of *Gatherer Components*.

4.2.3 Data Output

At the end of the HLT reconstruction chain, the HLT computes a trigger decision. The decision and additional payload is then sent to the DAQ via H-RORCs equipped with SIUs, which are situated in a subset of the FEP nodes, the so-called *HLTOUT* nodes. Only now the data blocks in the shared memory and their descriptors are removed. The

[7] A 64-bit number, see also section 5.5.1.1.

format of the HLT decision allows to enable or disable individual DDLs for storage [72]. Events or event fragments which are rejected by the HLT are then removed from the LDC buffers before the event building step.

Additionally to the output to the DAQ, a specialized component, the *TCPDump-Subscriber* (TDS), is used to tap into the data stream at any point and to provide access to the data blocks through a TCP socket. This is not only used to deliver reconstructed events for event displays and data quality monitoring but also to help developers to understand their code or even their detector at a very early stage of data-taking.

4.3 Data Reconstruction and Analysis

Following the hierarchical approach, the actual physics data handling is decoupled from the data transport as described above. The *HLT Analysis Framework* implements the reconstruction and analysis software, which is organized in modules, so-called *HLT Components*. This is part of the C++[8] based ALICE off-line software *AliRoot* (ALIce ROOT) [73–75] and can be used in both the off-line and HLT on-line environment. A detailed description of the HLT Analysis Framework and of its off-line integration can be found at [76]. Some relevant parts are briefly discussed here later.

4.3.1 Massive Parallelization

The throughput of the HLT is governed by the processing time of the different reconstruction components. In order to enhance the processing speed, the HLT reconstruction chain uses means of parallelization at all levels. It allows not only the components to process the event fragments for every detector in parallel following the detector layout, but also for parallelization inside one component. Here, the parallelism in the data structures themselves is used and, therefore, also the concurrency in the pattern recognition. Two complementary concepts are implemented in the HLT.

CPUs[9] in nowadays computers have registers as large as 128 or 256 bit. However, the actual data types used in the programming languages are at least a factor of 2 or 4 smaller, e. g. a floating point number (float) has normally 32 bit. Traditionally one register holds one value and one instruction operates on two registers in the *Single Instruction Single Data* (SISD) approach. Filling the empty space in the registers with the means of data vectors in the *Single Instruction Multiple Data* (SIMD) approach [77, 78], one operation can be executed on several values at the same time, as described in Figure 4.4.

Following this approach, the next level is the use of GPUs[10], which provide several hundreds of parallel processing units and, therefore, a greater speed-up of reconstruction time. Due to the large overhead of loading the data of each event into a GPU, their usage is only justified for very processing intensive tasks. Details can be found in [79, 80].

[8] C++ object oriented programming language
[9] Central Processing Units
[10] Graphics Processing Units

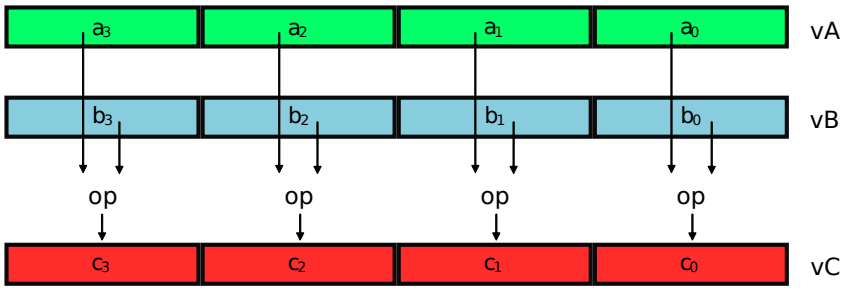

Figure 4.4: In the *Single Instruction Multiple Data* (SIMD) approach vectors of e. g. floating point values are stored as vectors `vA` and `vB` in the CPU registers. In that way, one operation `op` is applied on all values in the two registers at the same time and the result is written to `vC`.

4.3.2 Reconstruction Strategies

The on-line reconstruction in the HLT has to implement similar steps as the off-line reconstruction after the data-taking. However, due to the tight constrains of the reconstruction time, on-line reconstruction algorithms can be different from their off-line versions. Modified versions of some algorithms are justified as the reconstructed event has to be precise enough only for computing a trigger decision and not for off-line analysis. Additionally, not all required calibration parameters are available exactly at on-line reconstruction time, which again can lead to slightly varying results. As mentioned above, the HLT reconstruction follows strictly the detector hierarchy.

First Level Reconstruction The first level of reconstruction includes the unpacking of the compact raw detector data, the extraction of hit information and the computation of space-points, the so-called cluster finding. This is parallelized on the level of the detector read-out partitions (the incoming DDLs), in order to keep these large data blocks on the FEP nodes and to send only space-points, a much smaller volume of data, to the next reconstruction level. As this is very well suited for processing in hardware, the finding of 3D space-points from the TPC has as well been implemented for the FPGA of the H-RORC (see also [81]), which allows for faster processing at this step.

Second Level Reconstruction In the second level of reconstruction already several read-out partitions are combined, e. g. creating clusters of deposited energy in the calorimeters is done on the module level. Also the track finding in the TPC works on the sector level, combining the 6 read-out partitions within the same azimuthal range. Afterwards, all the tracking output of the 36 TPC sectors is merged into a single instance. These tracks are then propagated to the ITS and TRD using their reconstructed space-points. Also the initial interaction point, the primary vertex, is reconstructed using two different methods: only SPD space-points and combined tracks.

Event Merging The reconstructed information of the different detectors is finally combined, e. g. calorimeter clusters are geometrically matched to combined tracks, tracks from the muon arm are matched to ITS clusters, and event information from the *V0* and ZDC is added. All information is filled in an *Event Summary Data* (ESD) object, implemented in the `AliESDEvent` class which is used in both on-line and off-line processing. This gives the possibility to use the events reconstructed by the HLT also later in the off-line data analysis and, moreover, off-line analysis code can be executed inside the HLT.

4.3.3 Detector Calibration

An additional complication arises from the fact that particle detectors are macroscopic devices which have changing parameters. They can be misaligned with respect to each other or even their own inside structures deviate slightly from the ideal design geometry. Voltages, pressures, temperatures, or gas compositions can change over time and, therefore, alter the detector performance. This information and the related corrections are stored in the *Off-line Conditions DataBase* (OCDB) and provided for the off-line reconstruction. However, not all of these parameters are available at run-time for the HLT. They can be divided into two categories, leading to two different on-line calibration strategies.

Slowly Varying Parameters Slowly varying parameters are stable for a longer period of time, e. g. several runs. Here one can list detector alignment, inactive detector parts, voltages of the FEE, as well as gain calibration of read-out channels or coarse baseline temperature and pressure values. These parameters, a subset of the OCDB, are stored in the *HLT Conditions DataBase* (HCDB) and updated before start of the data-taking, so that they can be used by the on-line reconstruction algorithms.

Quickly Varying Parameters In contrast, some parameters, e. g. temperature and pressure, can vary on a rather short timescale. These values are extracted from DCS databases and the calibration parameters are computed on-line. As there are only milliseconds from the collision until the actual on-line reconstruction, these parameters can technically never arrive in time, so a prediction for them is stored in the HCDB. This can lead to a small discrepancy to the off-line reconstruction. However, this is negligible for the triggering process.

Calibration Algorithms The HLT is also a producer of calibration data. Special components are inserted into the reconstruction hierarchy producing calibration parameters on the basis of the reconstructed events. They are then not only used again for the on-line reconstruction, but can also serve as a baseline for the off-line reconstruction.

4.3.4 On-line Analysis

After and in between the reconstruction steps, analysis components can be attached, which allow for an analysis of the reconstructed events and, therefore, provide input for the data quality assessment and the trigger algorithms. An example for this is the reconstruction of secondary vertices from decays of the neutral strange K^0_s, Λ,

and $\overline{\Lambda}$ particles, the so-called *V0 decays*. The mean and width of the invariant mass distributions of the mother particles are a good estimator for the data quality.

Further tasks are the particle identification based on specific energy loss $\mathrm{d}E/\mathrm{d}x$ in the ITS and the TPC, as well as the electron identification using the transition radiation in the TRD, or even the estimation of the event-plane in heavy-ion collisions. Inspecting the un-like sign invariant mass distributions in the dimuon and dielectron channel prepares for the triggers as well as reconstructing high-p_{t} particles and jets.

Compression of raw and reconstructed data is another key task which is handled here. The data blocks are compressed with the help of lossless and lossy algorithms, depending on the necessary compression factor and tolerable physics impact. Reconstructed space-points of the TPC, e. g. , can be stored as distance to a reconstructed track [82], or raw data can be compressed using Huffmann Coding [83, 84].

4.3.5 Full Event Reconstruction

Unlike other high-level trigger systems in high-energy physics [85], high-level triggers in heavy-ion experiments fully reconstruct all read-out events [86].

This has clear benefits, but also some disadvantages. In so-called "stream-based" triggers, only parts of an event are reconstructed in parallel streams. If one of the streams has computed a positive or negative trigger decision, the processing can be stopped and computing resources can be spared, unlike for a full reconstruction. Also the latency of the events is higher than in stream-based triggers, as a full reconstruction needs more time, which then leads to a need of larger data buffers in preceding data flow layers.

The advantages of a full event reconstruction in heavy-ion physics as compared to high-energy physics can be explained by the different physics observables. While today's high-energy experiments are mainly focused on high-p_{t} probes (e. g. the ATLAS and CMS detectors even have an intrinsic minimum p_{t} by design), heavy-ion experiments need to measure the full phase space, to study the medium effects. Only fully reconstructed events allow to investigate bulk properties like the event-plane and the centrality, or even event-by-event particle ratios. The use of lower bound momentum filters, excluding particles from the search for secondary vertices, is another gain for physics trigger inputs. Additionally, the fully reconstructed event is very helpful for data quality checks, already during the data-taking as the off-line processing of heavy-ion events might not be finished soon after.

Summarizing, the full event reconstruction is the preferred tool for heavy-ion physics environments, as the benefits clearly outweigh the disadvantages.

4.4 Trigger

As discussed in section 3.2.1, the HLT is situated in the read-out chain after the detector read-out. Despite the name, the HLT trigger is more an event filter intelligently reducing the amount of read out data. Consequently, the final stage of the HLT on-line processing is the triggering step.

4.4.1 Trigger Framework

The HLT trigger framework allows for several trigger algorithms to run in parallel. Every of those algorithms produces an *HLT Trigger Decision*. The final *HLT Global Trigger Decision* is then a logical combination of their outputs and consists out of several parts :

TriggerName The name of the trigger following the ALICE and HLT conventions for trigger names.

Description A human readable description of the trigger algorithm.

Trigger domain The HLT data blocks to be read-out (e. g. an ESD or compressed data from a specific detector) and the list of DDLs to be read-out by the DAQ, the *HLT read-out list*.

The global trigger decision is logically merged in the *HLT GlobalTriggerComponent* and includes additionally the list of contributing trigger decisions and an array of trigger counters. Different trigger domains are combined with a logical or.

The HLT read-out list contains one bit for every DDL and implements the HLT decision for the DAQ. An event is rejected if all bits are disabled and accepted if all bits are enabled. Enabling only a subset of them describes the region-of-interest (ROI) read-out, which allows to store only relevant parts of an event. Disabling all but the HLT links, which behave like normal inputs on the DAQ side, offers the possibility do reject all the raw data, but store reconstructed or compressed data from the HLT. At the DAQ side the HLT read-out list is logical or'ed with the actual active DDLs.

Not all triggers need to be complex algorithms. The trigger framework also allows to define simple summary variables, like number of tracks, to be used directly in the global trigger.

The configuration of the trigger algorithms and the global trigger is done via the *HLT Trigger Menu*, which is identified by a unique name, the HLT TriggerMenuIdentifier. This identifier allows an easy selection of separate sets of running conditions by the ALICE shift crew. Every trigger menu selects the available trigger components, on which CTP trigger they should act, and a possible scale down. A more detailed description of the trigger and trigger menu naming schema can be found in appendix A. The configuration of the HLT and the trigger menu is discussed in detail in section 5.5.2.2.

4.4.2 Physics Triggers in the HLT

The HLT is capable to host a large variety of triggers. Simple triggers expecting one track passing a certain volume have already been used in the commissioning phase of ALICE and provided an effective trigger on cosmic particles. As its main purpose is the event filtering after the read-out, a key objective is the identification of rare probes such as dileptonic decays of Z^0, J/ψ, and Υ, as well as anti- and hyper-nuclei. A selection of these triggers is described here.

The TRD L1 trigger on dielectrons suffers from a rather high background of electrons from conversions and secondary decays, which can be cleaned in the HLT using the combined tracking information of the TPC and ITS. This can even be improved by the use of the specific energy loss $\mathrm{d}E/\mathrm{d}x$ in the TPC.

A good secondary vertex resolution and particle identification might allow for open charm and open beauty triggers. In order to reduce the possible combinatorics, first a

momentum filter is applied. Afterwards, the secondary vertices are calculated, using relaxed cuts compared to the off-line analysis, and only promising candidates are kept.

The L0 trigger of the muon spectrometer does not allow for a precise cut in p_{t}, as the muon trigger chambers are not sufficiently segmented. Here the HLT dimuon trigger removes the rather large background with help of tracking information from the slower muon tracking chambers, keeping only good μ^+ μ^- candidates.

Triggers on high-p_{t} particles help to extend the p_{t} reach of related studies and provide useful events for the detector calibration. High-p_{t} jets can be reconstructed and used for jet and di-jet triggers to study jet-quenching effects in heavy-ion collisions and to collect a baseline jet sample in pp collisions.

Photo-nuclear interactions occurring in ultra-peripheral heavy-ion collisions have to be separated from beam-gas interactions, as their signature looks similar in ZDC and V0 based hardware triggers.

A possible L1 TRD trigger on energy loss in the TRD can identify rare particles like heavy anti- and hyper-nuclei, but suffers from a huge background. Using the specific energy loss $\mathrm{d}E/\mathrm{d}x$ in the TPC these rare probes can be filtered out.

Furthermore, the identification of high-p_{t} cosmic particles in normal collision events is a valuable tool to help calibrating the detectors.

5. Commissioning of the HLT for pp collisions

The HLT is a complex prototype of a high performance computer with custom made software and partly even hardware. The assembly and commissioning of such a project is not just a push button operation, even if the HLT cluster itself is built out of commodity components. Being a prototype the commissioning was not and can never be a straight line process, but necessitated a detailed planning, which is laid out in section 5.1. However, lessons learned had to be taken into account, new ALICE developments had to be followed and sometimes even old decisions had to be revised. As the HLT has interfaces to almost all ALICE systems and is deeply embedded into the data stream, its commissioning is an important as well as challenging task.

This thesis is the main contribution to the commissioning of the ALICE HLT and its process is described in detail in this chapter. Starting with the introduction of the commissioning goals and its concept, it is followed by the achieved milestones, and the description of the hardware and software integration. The configuration of the HLT is a highly complex and challenging subject, which is discussed afterwards together with a description of the interfaces to other systems and the running procedure of the HLT. Finally, the most important commissioning part, the access to physics information is described.

As this is a rather technical chapter, a summary of the achievements of the HLT commissioning is given in this first section.

The high performance cluster of the HLT has been built up completely within the work of this thesis. It follows the *Beowulf* architecture, which is a cluster of commodity PCs, interconnected with a standard network, Linux or Unix based operating system, running open source software, and operated by a single head node. The HLT uses its own data-transport framework in order to ensure the distribution of the event data fragments to the processing entities.

The HLT cluster hardware consists of several components. 121 FEP nodes receive the read out detector data and send the HLT output via 470 optical fibers. Build-in FPGAs in these nodes perform the first level event reconstruction for the TPC. The further event reconstruction, as well as trigger and compression algorithms are executed on 51 computing nodes utilizing 408 processor cores. Moreover, several servers are used to provide the cluster infrastructure such as user management, network address management, distributed file system, as well as monitoring and steering applications. The single nodes are interconnected with 3 different networks: a Fast-Ethernet network for cluster monitoring and maintenance, a GigaBit-Ethernet network for the processed data, as well as a InfiniBand based backbone network, to overcome possible bottlenecks in the data network.

Every node, participating in the data-taking, needs a consistent set of application software. In order to ensure this, several methods have been investigated and considered.

Sequential copy mechanism and the distributed file system NFS don't scale with a large number of nodes. Treelike synchronization mechanisms needed to be developed at that time, that are able to cope with failures of some branch nodes and still reach every node inside the cluster. The distributed file system AFS is designed to keep a large number of small and medium files. Its architecture of writable data volumes and their read-only copies allow for a load-balanced reading of the stored files. This fits well to the needs of the HLT, where application software is only installed once and then read from thousands of data-processing components at the same time during start-up.

During the reconstruction thousands of processes are needed, to process the read out events of several hundreds of nodes. These processes have to be started and steered synchronously and have to be kept up-to-date with status informations and detector conditions. The configuration of these processes and their interplay with the interfaces to the outside are combined within the *RunControl*, which was developed in the context of this thesis. A hierarchy of so-called TaskManager processes orchestrates the single processing components and their provision with the needed data and information. The top-level entity is the so-called *RunManager*, which controls all TaskManagers and, therefore, is able to steer the whole HLT. He communicates via the HLT–ECS proxy interface to the steering system of the ALICE data-taking, the *Experiment Control System* (ECS).

The configuration of the HLT is completely determined by the HLT trigger menu. It defines which trigger algorithms are contributing to the HLT trigger decision and, therefore, also which detector data and reconstruction processes are needed. At the beginning of each data-taking the HLT and, therefore, also its TaskManagers, has to be reconfigured in order to allow for changes in the ALICE detector setup. In order to do this efficiently, a treelike structure of modular configuration objects was developed within this thesis, which allows to create the full configuration of the HLT. The single objects contain configuration parameters of the actual processing components, but also their dependencies with each other.

Interfaces to the other on-line and off-line systems are not only important for the data-processing itself, but also for the configuration of the HLT components. They can be divided in three classes according to the handled data types: physics data, configuration data and detector conditions data. In the course of the commissioning process all FEP nodes haven been installed in order to be able to connect all ALICE detectors to the HLT. The interfaces to the ECS, the *Detector Control System* (DCS), the *File eXchange Server* (FXS), and the *Off-line Conditions DataBase* (OCDB) provide all necessary configuration parameters as well as detector conditions needed for the functioning of the HLT.

Another major objective during the HLT commissioning, was the provision of access to on-line reconstructed events already during the data-taking, in order to support not only other detectors during their commissioning phase, but also allowed the HLT itself to study its performance. Moreover, by operating an on-line event display, the interactions could be visually inspected in real-time.

The access possibilities can be categorized in a primary data path via the DAQ and secondary data path, where the primary data path is the normal data flow of reconstructed, compressed, and/or triggered events. Via so-called *TCPDumpSubscribers* (TDSs), a possibility to tab directly into the data flow has been implemented, which allows to access all on-line produced information without disturbing the primary data

path. The TDS deliver synchronous or asynchronous data blocks in respect to the EventID, where the first ones contain all information of one event, which allows to visualize this information in an event display. The later ones can be histograms to be used in data quality monitoring.

5.1 Commissioning Concept

The commissioning of the HLT was structured in several phases, which were adapted to the general ALICE commissioning schedule.

Phase 1 In the first phase, the goal has been to build and commission a prototype of the HLT cluster, which was able to digest simulated data[1] and could be controlled from a single instance. Its realization was grouped in several steps.

1. **Hardware Setup** The initial step was the first setup of the cluster infrastructure in terms of network, server infrastructure, storage and cluster nodes.

2. **Software Infrastructure** The installation of a distributed file system, the user management and the cluster management system.

3. **Data Transportation** Commissioning and scalability tests of the data-transport framework on top of the cluster hardware for simple configurations, using dummy test data.

4. **Data Reconstruction** Integration of the reconstruction and the data-transport frameworks using simulated physics data and simple reconstruction components.

5. **Endurance Running** Setup of the on-line event display and endurance tests for configurations with increasing complexity.

Phase 2 The setup of the interfaces to the other on-line systems and the integration into the ALICE data-taking setup determined the second phase of the commissioning. Here, the main detectors have been integrated and raw data coming from them was processed.

1. **Hardware Interconnections** The network connections as a basis for the interfaces to the other on-line and off-line systems had to be setup and commissioned. This included the Ethernet connections as well as the optical-fiber connections for the data in- and output.

2. **Detector Integration** One after the other, the main detectors were added, while testing their data input and the first reconstruction steps.

3. **Interfaces** Receiving and sending of conditions data was put in place as well as the receiving of configuration data. Commissioning of the on-line data access for detector experts.

[1] The data was simulated with the AliRoot framework, details concerning data simulation can be found here [73, 74].

4. **RunControl** Integration of the configuration interfaces and the configuration of the data-transport framework to the RunControl entity, including the setup of the operator workstation in the ALICE control room.

5. **Endurance Running** Data reconstruction in the first ALICE data-taking campaign of cosmic particles. Raw detector data were reconstructed and displayed on the on-line event display.

Phase 3 In the third phase, the main objectives have been set to the full integration of the HLT into the data path, the adaption of the ALICE configuration model and a hardware extension.

1. **Data Output** The commissioning of the data sending hard- and software on the HLT side and data receiving hard- and software on the DAQ side. Setup of on-line data access for the whole collaboration.

2. **Full Reconstruction** Combining reconstruction output of all detectors and transmitting it to permanent storage via the DAQ and to the on-line event display.

3. **Trigger Framework** Integration of the trigger framework into the HLT data path and sending decisions to the DAQ.

4. **ALICE Configuration** Adaption and commissioning of the top-down configuration approach via the *ALICE Configuration Tool* (ACT), where the HLT entity is completely defined by two configuration parameters, a trigger menu identifier and the DAQ mode.

5. **Hardware Extension** Expansion of the computing power and network throughput for the next commissioning phase.

Phase 4 The final commissioning phase was determined by endurance benchmarks of complex configurations, tests of hardware accelerators and physics data-taking.

1. **Data Analysis** Integration of data-quality and data-analysis components into the reconstruction and into the DAQ data quality monitoring system.

2. **Endurance Running** Commissioning of the stability of the whole system.

3. **High-Speed Running** Adaption of the HLT reconstruction to the detector read-out rate, enlarging the number of reconstruction processes.

4. **Data Taking** Finally, the full integration of the HLT into the ALICE running environment and continuous physics data-taking.

5.2 Commissioning Milestones

Early in the year 2006, the commissioning of the HLT at CERN started. Here, a short time-line with the goals reached is given. A more detailed description can be found in the following sections.

The Beginning – Spring 2006

The commissioning began with the installation of the first network cables into the floor of the *Counting Room 2* (CR2) and *Counting Room 3* (CR3), which are described in detail in section 5.3.

TPC Commissioning – Spring/Summer 2006

With the start of the commissioning of the TPC in spring 2006, the first HLT PCs with H-RORCs have been brought to CERN. They have been connected at the surface to the DAQ and, therefore, also to the TPC. After an initial commissioning of the H-RORCs and the HLT software, the first raw TPC data events have been read-out into the HLT, which can be seen in Figure 5.1 (see also [87]). At that stage, the first level reconstruction software, including zero-suppression and cluster-finding algorithms could be tested. Very soon after, in summer 2006, the PCs have been installed in CR2, establishing the basis of the HLT cluster.

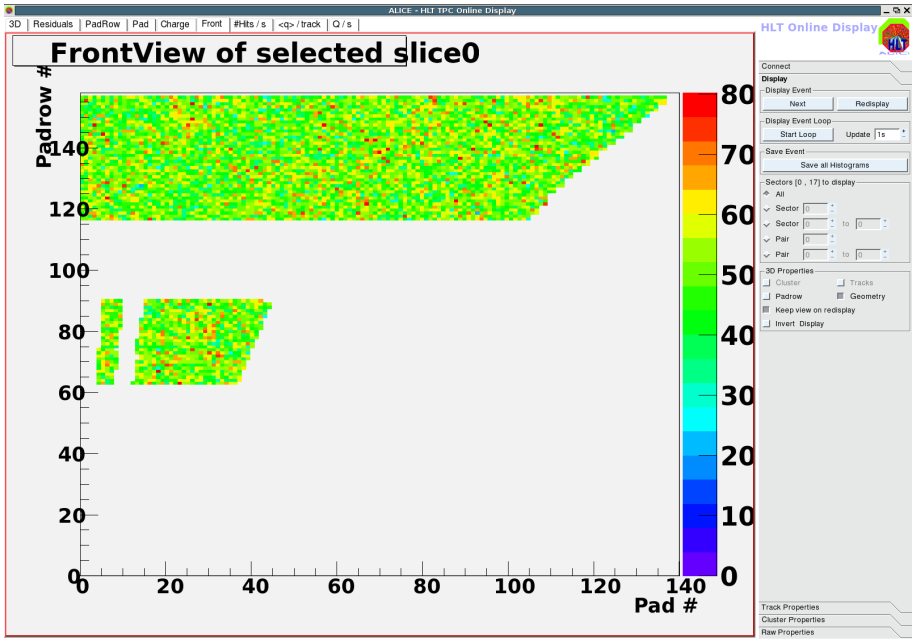

Figure 5.1: First data from the TPC read into the HLT and displayed in the HLT TPC event display, showing non zero-suppressed raw data of two and a half read-out partitions of one sector. In the half read-out partition two missing TPC front-end cards can be identified. The pad number is shown on the x-axis, the pad-row number is shown on the y-axis, and the z-axis is summed charge of the pad.

HLT Cluster Assembly – 2007

With the installation of the GigaBit-Ethernet network, the infrastructure machines, and the first 80 *Front-End Processor* (FEP) nodes the building of the HLT cluster started in spring 2007, as shown in Figure 5.2. With the basic installations of operating system, distributed file system and HLT software the first large scale tests were done. Naturally,

at the same time the SysMES cluster management system was deployed on the cluster and started with simple automated but necessary controlling and reaction tasks. Later in the summer, the second batch of FEP nodes followed.

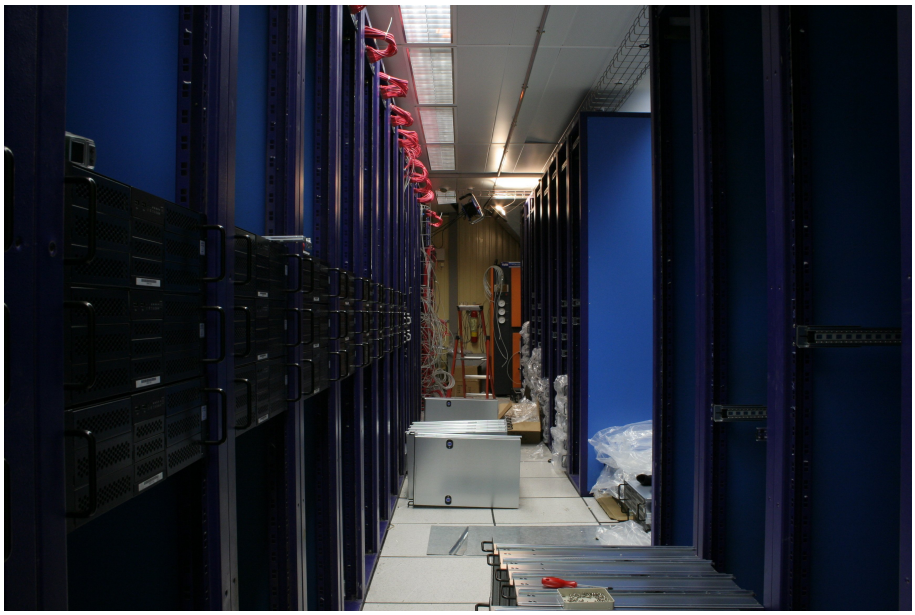

Figure 5.2: Installation of the first batch of *Front-End Processor* nodes into CR2.

In the course of simplification for the HLT users, the *RunControl* framework was developed and deployed as a front-end to the data-transport framework. In order to allow the HLT to take part also in the commissioning of the other ALICE detectors, several so-called *operator* users have been created. Each of them represents an HLT instance, which can be operated independent and in parallel to the others. Now, for the first time, all TPC sectors have been read-out and processed in the HLT. Additionally, the HLT commissioning of the TRD, PHOS, and the MUON-Arm has been started.

As other on-line systems also became available, the interfaces to them and the off-line systems have been put in place and their commissioning has been started. The final step for this phase was done in December 2007, when ALICE started their first combined efforts to run not every detector stand-alone in parallel, but together as one experiment. With the first data from cosmic particles the interplay of the different reconstruction steps in the HLT could be exercised.

ALICE Commissioning with Cosmic Particles – 2008, 2009

In expectation of pp collisions in late 2008, ALICE including the HLT started a cosmic data-taking campaign in spring 2008 to align and calibrate the detectors. The HLT used this campaign to test extensively the interplay of the reconstruction of the different detectors and commissioned the *HLTOUT*, the output to the DAQ. An example event of a cosmic particle is shown in Figure 5.3. With the three ITS detectors added to the HLT the vertex finding and combined ITS + TPC tracking became available. In summer the missing FEP nodes to connect all ALICE detectors and the first 15 *Computing Nodes* (CNs) were included. Additionally, a small development cluster has been built up in CR3.

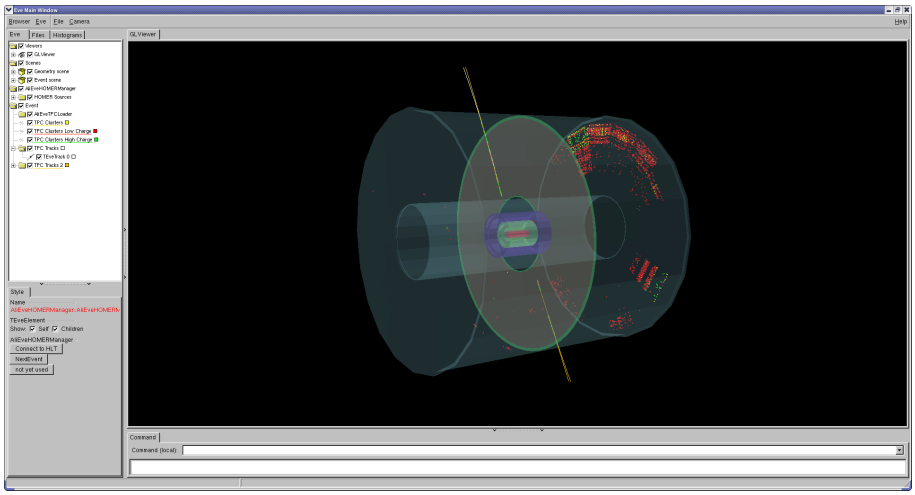

Figure 5.3: A cosmic particle is traversing the TPC. The reconstructed track is light brown and the reconstructed clusters are color-coded from green to red, depending on their charge. On the right end-cap of the TPC, electronic noise can be seen.

Close to the anticipated time for the first pp collisions, compression of SDD raw data was implemented and deployed. For the injection tests of beam into the LHC and the first circulating beams, the HLT was operated in triggering mode, rejecting all direct SDD raw data but sending compressed SDD data from the HLT. The full HLT reconstruction was tested to run at a data rate of 900 Hz.

During a powering test of the LHC magnets in sector 3-4 on September 19^{th}, 2008, a dipole quench occurred causing a faulty electrical connection to break. This led to a release of 15 t cold supra-fluid Helium and to mechanical damage on a third of the sector's magnets [88].

As a consequence of the incident, additional time became available, which the HLT used to start a re-commissioning of the cluster hard- and software, after an assessment of the previous running periods. The cluster infrastructure was restructured and the software algorithms revised, as well as the EMCAL detector included. Additional CN nodes, as well as the backbone InfiniBand network, have been installed to complete the pp setup of the HLT in late spring 2009.

Trigger Commissioning of the HLT– Autumn 2009

In summer 2009, the ALICE cosmic data-taking campaign restarted and the HLT deployed and commissioned the triggering framework. Towards September, the cosmic L0 triggers were opened to a read-out rate of $\approx$ 200 Hz and the HLT was operated in triggering mode to select only events with cosmic particles traversing the ITS. An example of these events can be seen in Figure 5.4. With the LHC injection tests, now also the calorimeters could be tested with particles within the HLT, as it can be seen in Figure 5.5.

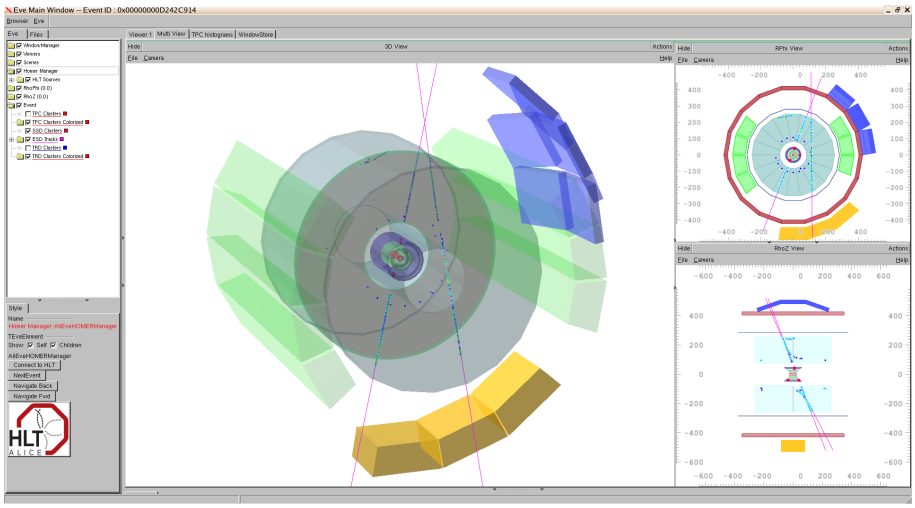

Figure 5.4: Two high-p_t particles traversing the TPC and ITS indicated by the pink reconstructed tracks. The clusters in the TPC are color-coded from blue to red, depending on their charge. SSD clusters are colored red.

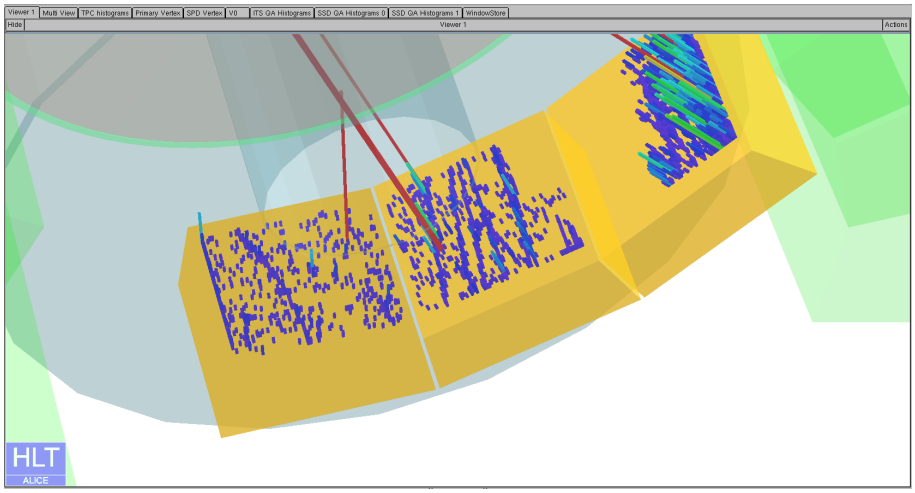

Figure 5.5: Tracks of high energy muons, formed by energy deposit in the crystals, can be seen inside the PHOS calorimeter, traversing it parallel to the beam direction. They have been produced, while the LHC beam was dumped on a collimator in front of ALICE at the end of the TI2 injection line from the SPS. The large red spikes indicate noise in the read-out electronics.

First pp Collisions at $\sqrt{s} = 900\,\text{GeV}$ – November 23rd, 2009

For the first run with pp collisons at $\sqrt{s} = 900\,\text{GeV}$ in ALICE on November 23rd, 2009 no *stable beam*[2] was declared and so only the ITS, the calorimeters and the V0 detector were turned on. The on-line reconstructed vertex based on reconstructed SPD clusters of the HLT (see Figure 5.6) confirmed immediately the presence of pp collisions. A data rate of $\approx 1\,\text{Hz}$ was reached, limited by the interaction rate.

[2] *stable beam* is a state of the LHC, announcing stable and, therefore, safe conditions for physics data-taking.

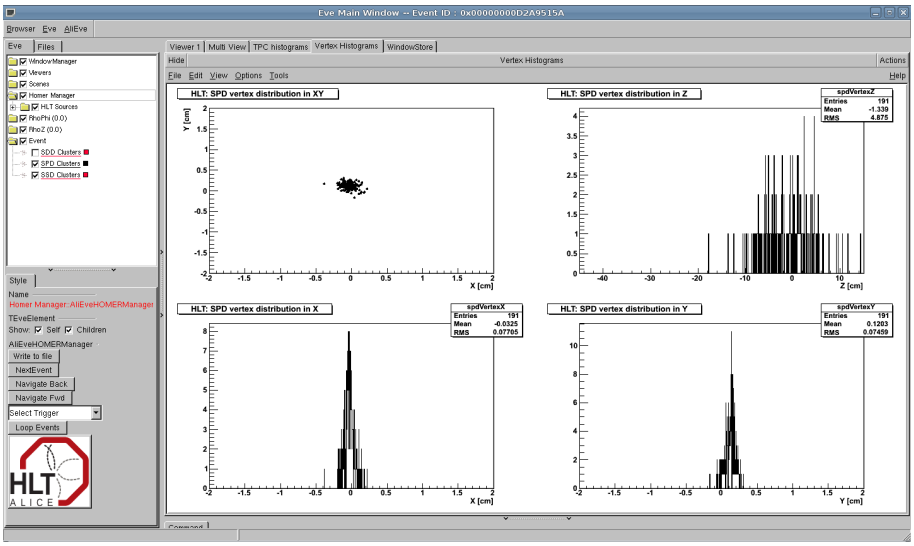

Figure 5.6: The on-line reconstructed primary vertex based on SPD clusters for run 101498. The figure shows, counter-clockwise from top left, the position in the transverse plane for all events with a reconstructed vertex, the projections along the transverse coordinates x and y, and the distribution along the beam line (z-axis). From [89].

First pp Collisions with full ALICE at $\sqrt{s} = 900\,\text{GeV}$ – December 6^{th}, 2009

In the early morning of December 6^{th}, 2009, *stable beam* was declared by the LHC. ALICE and, therefore, also the HLT started data-taking with all detectors turned on. An event display of one of the first on-line reconstructed pp collisions in the TPC can be seen in Figure 5.7. With the increased data rate over the next days neutral strange particles from secondary vertex decays could be seen on-line as shown in Figure 5.8.

First pp Collisions at $\sqrt{s} = 7\,\text{TeV}$ – March 30^{th}, 2010

With the first pp collisions at $\sqrt{s} = 7\,\text{TeV}$ on March 30^{th}, 2010, a major part of the commissioning for the HLT has been completed. An impression of one of the first collisions can be seen in Figure 5.9. The small interaction and read-out rate at the beginning soon turned into several kHz collision rate and 900 Hz read-out rate, respectively. In this first phase of the data-taking, the HLT was only used for on-line monitoring and provided an event display.

5.3 Hardware Integration

A modern HLT is a scalable high-performance compute cluster with high availability but low purchasing and operating costs. This is matched by the so-called *Beowulf* architecture [91, 92] and consequently the ALICE HLT is built as such. A *Beowulf* is a cluster of commodity PCs, interconnected with a standard network, Linux or Unix based operating system, running open source software and operated by a single head node [93]. However, as opposed to the original *Beowulf* clusters the software for parallel

[3] An Armenteros-Podolanski plot is a two dimensional plot, which is used in the analysis of two-body V0-decays [90].

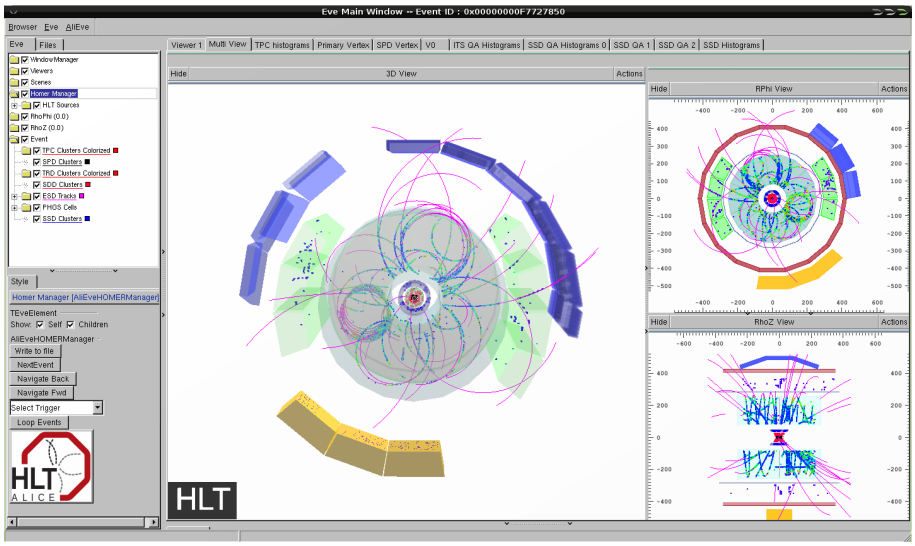

Figure 5.7: One of the first pp collision at $\sqrt{s} = 900\,\text{GeV}$ as seen in the HLT on-line event display. Reconstructed tracks are pink. The clusters in the TPC are color-coded from blue to red, depending on their charge. SPD clusters are colored black, SDD clusters are colored red, SSD clusters are colored blue, and TRD clusters are colored blue.

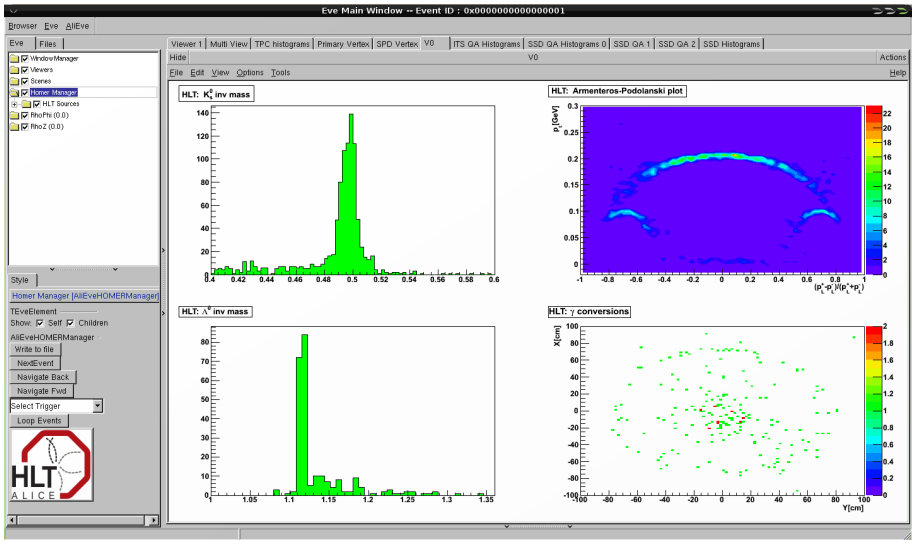

Figure 5.8: On-line reconstructed neutral particles from secondary vertices. The figure shows, counter-clockwise from top left, the invariant mass of K^0_s, the invariant mass of Λ and $\overline{\Lambda}$, the conversion points of γ conversions in the transverse plane, and the Armenteros-Podolanski plot[3].

processing in the HLT is not MPI[4] based, but uses its own data-transport framework as described in section 4.2.2.

The building and commissioning of such a compute cluster needs to be carefully planned and carried out. It is inevitable for its construction to have a detailed planning of

[4]Message Passing Interface

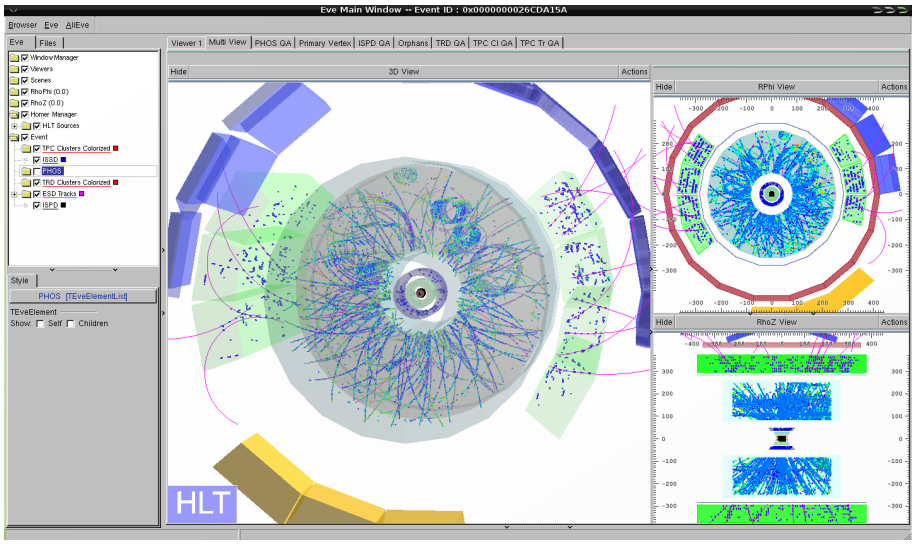

Figure 5.9: One of the first pp collision at $\sqrt{s} = 7\,\text{TeV}$ as seen be the HLT on-line event display. Reconstructed tracks are pink. The clusters in the TPC are color-coded from blue to red, depending on their charge. SPD clusters are colored black, SSD clusters are colored blue, and TRD clusters are colored blue.

the routing of the network infrastructure before [94]. Additionally, the layout of the head nodes, interface nodes and storage servers needs to be defined as well. Furthermore, it is very important for real-time applications to foresee not only the production environment, but also an equally equipped test and development environment [95].

5.3.1 P2 - Counting Room Layout

The HLT cluster itself is situated in two containers hanging in the shaft to the cavern of the ALICE detector at the LHC interaction point 2 (P2). These so-called *Counting Rooms* (CRs) are equipped with standard 19-inch CERN racks.

Counting Room 2 *Counting Room 2* (CR2), the upper one, belongs solely to the HLT and houses 40 racks in 3 rows (X, Y and Z), where one rack (X1) is reserved for the CERN electricity service (TS-EL). An overview of the layout is given in Figure 5.10, where each rack is associated to a given ALICE sub-detector, as indicated. Four racks (Y2, Y3, Y4 and Y8) are dedicated server racks and are connected to a UPS[5] system, shared with the DCS system. This battery based UPS was tested to deliver power for at least 20 minutes after a power-cut, enough to safely shutdown the HLT server nodes. The non-infrastructure nodes are connected to normal power, as they are build up of standard non-vital components. In case of interruptions of the ALICE cooling circuits, an emergency procedure is in place, where automatically tap water is used to keep the water flow alive, and therefore, also the rack cooling.

Counting Room 3 *Counting Room 3* (CR2), the lower one, is shared by HLT and DCS. The X row with 12 racks belongs to the HLT and the rows Y and Z with a total

[5]Uninterruptible Power Supply

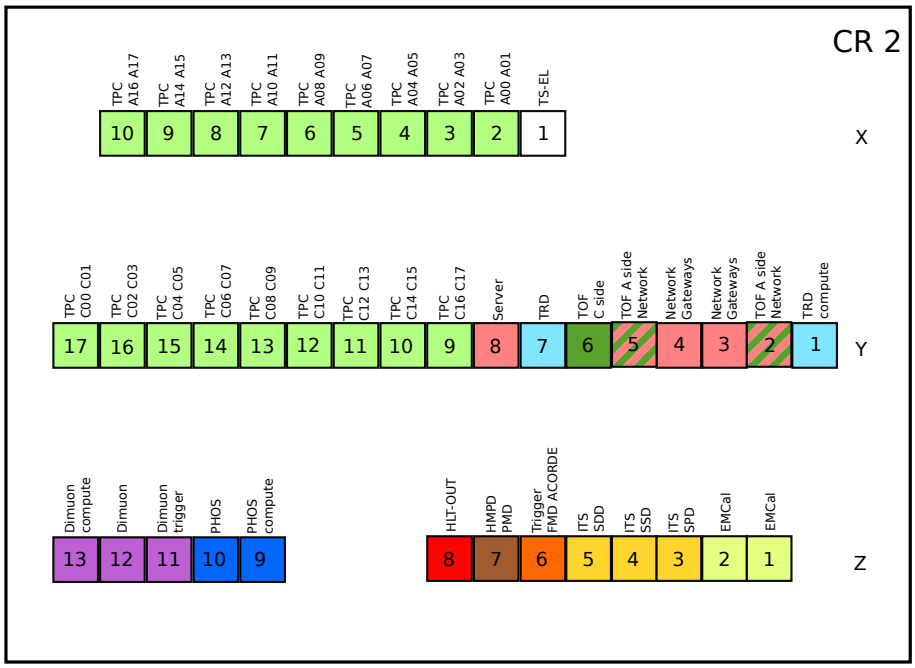

Figure 5.10: Counting Room 2 (CR2) is made up of 40 racks distributed in 3 rows (X,Y and Z), where the rack X1 is reserved for the CERN electricity service (TS-EL). The cluster infrastructure nodes are distributed in the UPS powered racks Y2, Y3, Y4 and Y8. Every rack is associated to an ALICE detector as indicated on the drawing.

of 23 racks belong to the DCS. The HLT racks were foreseen for a later extension of the HLT. However, with today's packing density of CPUs per height unit, this is very unlikely. Therefore, the racks X11 and X12 have been used to install a development cluster.

19-inch Racks The racks used in both CRs are standard CERN 19-inch racks (type `LEP-56PC`). Unfortunately, they don't comply completely to the EIA/ECA-310 standard [96], so the rack mounting rails of standard commodity rack-mountable PCs do not fit out of the box. The rails have to undergo a time-consuming procedure to make them fit. In order to cool the equipment in the racks, they are equipped with a water cooled heat exchanger and 3 fans in the back door.

Each rack is subdivided in 56 height units, so-called rack units (Us)[6]. Switches are normally 1 U high, nodes reach from 1 U to 4 U. In general, it is foreseen to put a maximum of 17 3 U machines in one rack, to be able to provide enough power and cooling. The power in a rack is provided via a power distribution unit, installed in the back half of the four topmost height units.

Network Infrastructure Each rack has been equipped with a 24-port Ethernet patch-panels at position −1 U. Every Ethernet connection of the nodes in a rack goes via the patch-panel. The patch-panels have been connected with Ethernet cables in the false floor to corresponding patch-panels in the racks Y3 and Y4, to allow a central place

[6] 1 U = 44.45 mm, defined by the EIA/ECA-310 standard[96, 97]

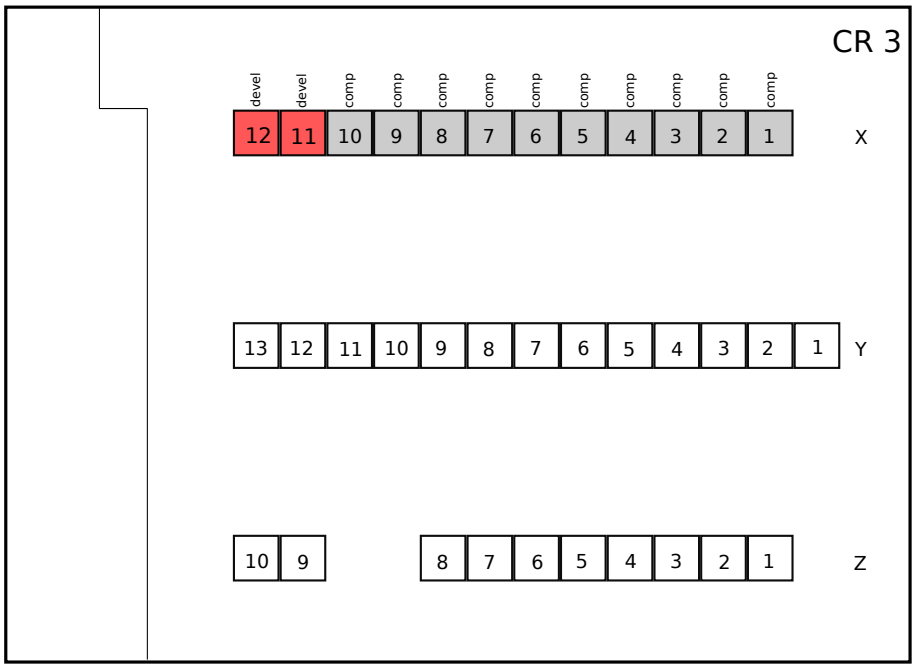

Figure 5.11: Counting Room 3 (CR3) is divided between HLT and DCS, where the X row with 12 racks belongs to the HLT. The development cluster is situated in the racks X11 and X12. All other racks (X1 – X10) are currently empty and foreseen for a future extension of the HLT.

for the network switches in the racks Y2 – Y5. This decision had been taken, as back then, it was not clear yet if one big switch or hierarchy of switches would be used. A schematic drawing of the patch-panels is shown in Figure 5.12.

5.3.2 Network

The HLT uses the IP[7] protocol stack for the inter-node communication. Its network is a sub-network of the CERN technical network and has been realized as a class-B[8] network in the 10.162.x.x domain. The IP address assignment can be found in detail in appendix B. The network range is divided in 4 sub-networks, from which two are the general purpose GigaBit-Ethernet networks for data-transfer, one is a Fast-Ethernet maintenance network and one is the InfiniBand backbone network.

Maintenance Network – Fast-Ethernet Every rack hosts a 24-port Fast-Ethernet maintenance switch in the uppermost possible rack position. This so-called *charm switch*[9] is connected to the ports of the management interfaces of the installed nodes in that rack. The up-link goes via port 24 of the patch-panel and all maintenance switches are combined in 48-port switch (*SWCHARM*).

[7] Internet Protocol, RFC 791 [98]

[8] A class B network has an address range allowing for 65,354 hosts. Every IP address starts with the bits `10` followed by 14 bits for the network and 16 bits for the hosts [99].

[9] At the time of naming, solely CHARM cards [100] where used as management interfaces, later also other devices have been added, see also appendix D.1.

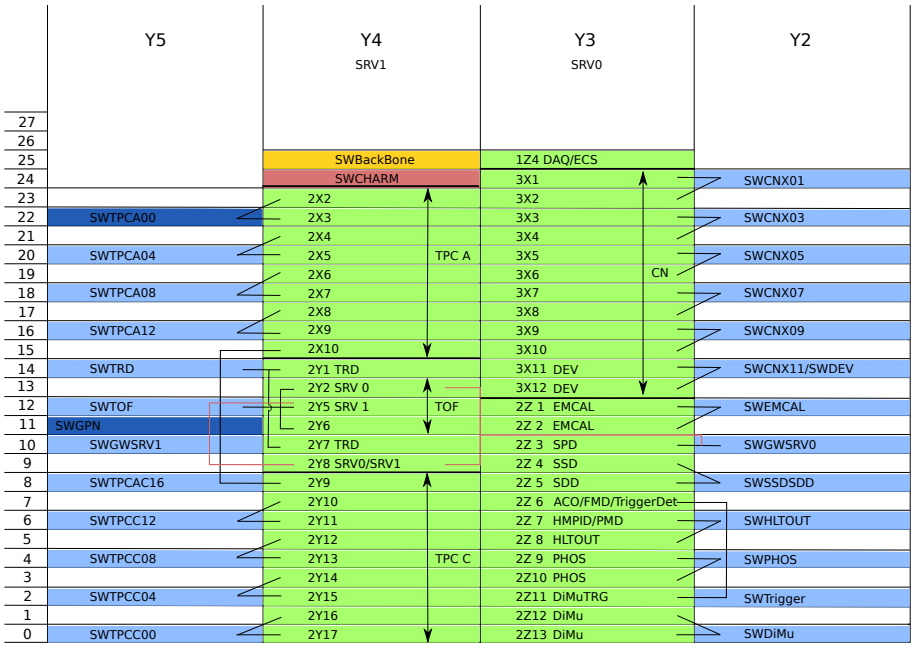

Figure 5.12: Layout of the network racks Y2 – Y5 in CR2, showing the 24-port Ethernet patch-panels in green and their associated 48-port GigaBit-Ethernet switches in blue. Colored lines from the patch-panels to the switches indicate their connection (black for normal switches and red for server switches). Port 24 of every patch-panel is connected to the management switch SWCHARM (red), which itself is connected via its port 48 to the switch SWGWSRV1. The up-link ports 47 and 48 of every switch is connected to the backbone switch SWBACKBONE (orange). A connection to the DAQ network is provided via the patch-panel 1Z4 and the DCS network can be reached via the patch-panel 3X6.

Data Network – GigaBit-Ethernet The HLT follows a hierarchical approach and uses the concepts of data locality. A treelike data network structure has been designed and built to realize this concept. Two racks are combined in one 48-port switch, using 23 patch-panel ports each (Schematically shown in Figure 5.13 and Figure 5.12). The two remaining switch ports are the up-link, where all GigaBit-Ethernet switches are combined in a 48-port backbone switch (*SWBACKBONE*).

This design uses the advantage, that the largest data blocks are processed within one switch and only smaller blocks cross the switch boundaries. As only a maximum of 17 3 U nodes can be physically installed in one rack, 6 patch-panel ports are not assigned and can be used for trunking of the second Ethernet port of selected nodes. An overview of the switches and the patch-panels can be seen in Figure 5.12.

Backbone Network – InfiniBand A potential bottleneck of this approach is the GigaBit-Ethernet backbone switch. To overcome this, a high throughput QDR InfiniBand[10] (IB) [102] backbone network has been put in place, which is laid out in Figure 5.14. The IB network again has a treelike structure, using four 36-port switches. The top-level switch (switch D in Figure 5.14) connects with 12 ports each to the three

[10] Quad Data Rate InfiniBand, allows a maximum data-rate of 40 Gbit/s.

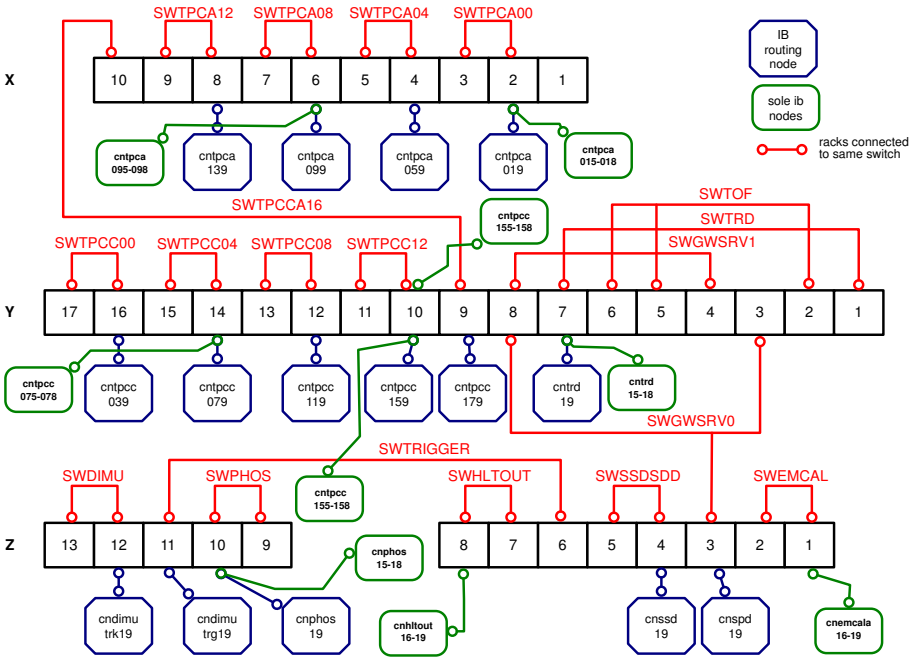

Figure 5.13: Overview of the GigaBit-Ethernet network. Switches are indicated as red connections and InfiniBand (IB) routing nodes are shown as blue boxes. Green boxes depict sole IB nodes. Adapted from [101].

switches lower level switches (A, B, and C), providing half bi-sectional bandwidth.

One node in every GigaBit-Ethernet switch, the so-called IB-routing node (shown as blue boxes in Figure 5.13), is connected to the backbone network, to allow for high throughput inter-switch communication. The infrastructure nodes and additional computing nodes (shown as green boxes in Figure 5.13) are connected to both the GigaBit-Ethernet and the InfiniBand network.

Data Network – Optical Fibers Physics data, as well as the HLT decisions, are exchanged via 470 optical fibers between the DAQ and the HLT networks. The endpoints of the inter counting room connections are manifested as optical patch-panels in the front half of the topmost rack positions. Their distribution over the different racks is described in appendix C.

5.3.3 Nodes

The HLT cluster is split in three groups of nodes. All service, storage and server machines are combined in the infrastructure group. Data processing nodes are built out of the group of computing nodes and the group of Front-End Processor nodes.

In order to follow the massive parallelization schema of the HLT, all processing nodes are multi-cpu, multi-core machines, with up to 2 CPUs and 8 cores per node, as well as up to 3 GByte RAM[11] per core. This design decision of multi-core nodes together with the HLT data-transport framework, allows to maximize the memory utilization of one node, and hence minimizes the network traffic between the nodes. As described in

[11] Random Access Memory

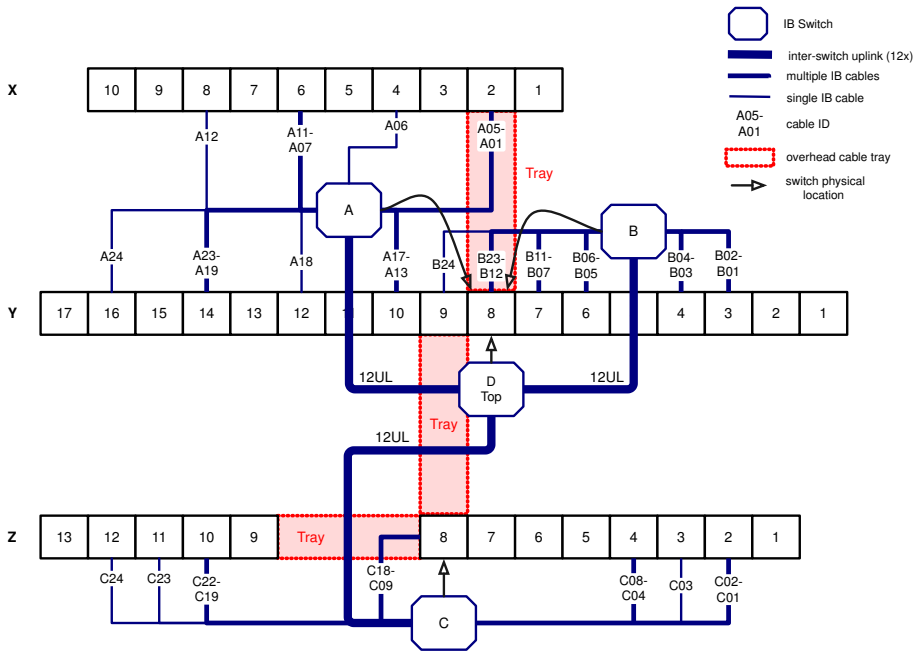

Figure 5.14: Overview of the InfiniBand backbone network hierarchy. From [101].

section 4.2.2, several processing components on the same node need no extra copying step of the data, but inter-node communication does.

Front-End Processor Nodes

The *Front-End Processor* (FEP) nodes are standard rack mountable PCs, which serve as the interface for physics data. Each node is equipped with two H-RORCs, allowing for a maximum of four optical-fiber connections. A total of 460 incoming and 10 outgoing DDLs are attached to 121 FEP nodes, which is the final setup for all ALICE sub-detectors. A detailed description of the nodes and there equipment can be found in appendix D.1.1. The FEP nodes are distributed to the different racks following the DDL schema of their appointed sub-detectors as listed in Table D.1. A maximum of five nodes per rack allows the use of the GigaBit-Ethernet trunking ports in the patch-panel. However, due to incompatibilities of GigaBit-Ethernet trunking with the InfiniBand setup, the trunking was abandoned at a later stage in the commissioning.

FEP nodes are situated in the first layer in the HLT processing hierarchy, and, as so-called HLTOUT nodes, also in the last stage before the processed data is sent to the DAQ. Therefore, parts of their main memory is used as front-end, respectively hlt-out buffers. These buffers are realized as BigPhysArea [103] shared memory, which is a large consecutive memory block, allocated at boot-up of the node[12]. The H-RORCs write to, respectively read from these buffers.

As the front-end buffers keep all events until they are sent out, they must be sufficiently large to buffer incoming events in case of an event which needs a much longer processing time than the average event, or even a temporary failure inside the processing chain. Therefore, the buffer size in the configuration for pp data-taking is

[12]Normal UNIX shared memory does not allow to allocate such a large consecutive block.

set to 1 GByte per incoming DDL link. With an average TPC event size per read-out partition of 2.7 kB (average event size 580 kByte / 216 partitions, see section 3.1.1), the buffer can hold up to 370,370 events, corresponding to 370 s at read-out rate of 1 kHz. At the other end of the processing chain, the hlt-out buffers need to cope with back pressure from the DAQ. They are configured to be 500 MByte per outgoing DDL link, due to a larger event size for outgoing events (see section 5.5.1.1).

Additional 3 GByte of `System V`[13] shared memory [71] is reserved on every node. This memory is used for the exchange of data blocks between processing components, as described in section 4.2.2.

Computing Nodes

The *Computing Nodes* (CNs) form the bulk of the HLT processing infrastructure. However, for the data-taking with pp collisions only 40% of the final setup has been deployed, providing sufficient computing power for the first year. A total of 51 CN nodes have been installed, made up of two generations, providing 408 processing cores. A detailed description of the hardware can be found in appendix D.1.2. More computing power is only needed for heavy-ion collisions, for which the number of CN nodes can be easily extended.

All CN nodes are directly connected to the IB backbone network, depicted as green and blue boxes in Figure 5.13. This allows for a higher network throughput for the inter-detector data exchange, as they are mainly used in the HLT processing stages 3, 4, and 5, which are described in section 4.1.4.

The choice of the second generation of CN nodes was, amongst other reasons, mainly driven by their modular chassis design (see appendix D.1.2). The hot-swappable drawers can be easily exchanged in case of failure and even more, then can be left out and auxiliary hardware can be installed in the empty space. This was used for the installation of GPU cards, as an additional processing entity in the HLT reconstruction chain. During the pp commissioning phase, only one of those was installed, in order to exercise the interplay of GPU and data-transport network.

Infrastructure Nodes

The group of the infrastructure nodes comprises all non-processing nodes, which are used to support the functioning of the HLT and therefore, the ALICE data-taking. They can be arranged in functional subgroups (storage/database, gateway/portal, monitoring/gui, and development), which shall be described here. Details of the infrastructure node hardware can be found in appendix D.1.3.

High Availability / Fault Tolerance To allow for high availability, two redundant instances for all gateway, portal, gui, database, and monitoring nodes have been deployed. For the storage services, this is handled on the level of the distributed file system. Furthermore, no high availability is needed for the development nodes. The storage servers are all equipped with eight hard disks and a hardware RAID[14] 6[15]

[13] Linux standard shared memory

[14] Redundant Array of Independent Disks

[15] RAID 6 : Striping with dual parity, more details about different RAID levels can be found here [104].

controller, tolerating up to two hard disk failures at the same time.

The nodes are physically distributed in three racks, each with its own power distribution unit, connected to UPS power. Development and storage, as well as both instances of database and monitoring nodes are situated in Y8. The first instances of the gateway, portal, and gui servers are hosted in Y3 and the second ones in Y4. Additional redundancy is reached with two GigaBit-Ethernet switches (SWGWSRV0 and SWGWSRV1, as depicted in Figure 5.12 and Figure 5.13), where the first instance of the infrastructure nodes are connected to SWGWSRV0 and the second ones to SWGWSRV1.

Storage/Database Server Four *Mass Storage* (MS) and two *DataBase* (DB) nodes are equipped with hardware RAID 6 systems optimized for storage capacity and access speed respectively (see section D.1.3).

The MS servers are used to provide storage for the application software and special user directories. Additional storage space is installed on two development nodes (dev2 and dev3), which host only production uncritical information, like normal user directories and simulated and real test data used to benchmark and stress-test the reconstruction chain. In total, the raw storage capacity is 36 TByte, with a RAID 6 net-capacity of 27 TByte.

An oracle database for the SysMES management environment and several mysql databases are hosted on the DB nodes with a total of 4.8 TByte raw capacity and 3.6 TByte net RAID 6 space.

Gateway/Portal Server The hardware setup of the gateway and portal servers is described in appendix D.1.3 and their external connections are given in appendix D.3. User login from the CERN GPN[16] to the HLT cluster is realized via two *Gateway* (GW) nodes, details of the different access possibilities are laid out in appendix E. Furthermore, the GW nodes host several administrational services, which are discussed in section 5.4. Therefore, they don't allow direct access and the users are automatically forwarded to the development nodes.

Access to the HLT cluster is needed not only for users, but also for the interfaces. They can be classified by the network they need to access. Dedicated portal nodes to the DCS/ACR, DAQ/ECS and the GPN networks are implemented. The interfaces and their setup are discussed in detail in section 5.5.1.

Development Server Four *Development* (DEV) servers are provided for user testing and software compilation. They are used as login nodes for normal users and allow to access not only the internal development cluster, but also the production environment (FEP and CN nodes). Details about the hardware setup of the development servers are laid out in appendix D.1.3.

Monitoring/GUI Server The monitoring nodes are dedicated to the setup for SysMES and other monitoring applications. An overview of the HLT cluster and HLT reconstruction to the HLT operator is provided by the GUI[17] application hosted on the GUI

[16] General Purpose Network
[17] Graphical User Interface

servers. Details about the hardware setup of the monitoring and GUI servers can be found in appendix D.1.3.

5.4 Software Integration

The commissioning of a high-performance compute cluster does not only include the setup the hardware, but also the software. As the HLT was designed and built from scratch, this involves every system-administrator task from installing and updating of an operating system, setting up the network and a user administration, managing a distributed files system, monitoring the cluster infrastructure and network, till the creation of a backup system, and the managing of application software.

This section describes the key issues which were handled during the software commissioning, which naturally happened in parallel to the hardware commissioning. The selection process of the operating system is outlined together with the choice of a distributed file system, serving the application software packages. Finally, an overview of the application software as well as its organization is given.

5.4.1 Cluster Management

The HLT was designed to be as autonomous as possible. Even if all outside network connections to it would fail, the HLT should be still functioning. Therefore, the HLT cluster has been built with its own DHCP[18] and DNS[19] services, in order to assign IP addresses and hostnames to the cluster devices (Details can be found it appendix B). These services are fed by an LDAP[20] directory, which also serves as a basis to the HLT user management. Kerberos [105] has been chosen as network authentication protocol in order to provide better security for users on a distributed file system.

High availability and fault tolerance is an important aspect when managing a real-time system. For this reason all the above mentioned services have been deployed in a fail-over mode, one instance running on each gw0 and gw1 nodes.

5.4.1.1 Operating System

As a true *Beowulf* cluster, the HLT uses an Linux based operating system. Several commercial and an open-source Linux flavors exist on the market. The selection of an operating system for the high-performance HLT cluster has been driven by the following criteria:

- Easy maintainable, upgrade of operating system without its full re-installation
- Long term support for security upgrades, in order to keep a stable system for a longer period
- Support for cutting edge hardware, such as latest CPUs, motherboards, GPUs, and network technologies
- Non-commercial

[18] Dynamic Host Configuration Protocol
[19] Domain Name System
[20] Lightweight Directory Access Protocol

- Existing knowledge inside the HLT group, in order to allow knowledge transfer

A debian [106] based distribution was selected, due to its package management system, which provides the required flexibility when installing new software packages, software updates, as well as upgrading the operating system as such. As an attribute to the South-African part of the HLT group, **Ubuntu Server LTS**[21] [107] was chosen. At the beginning of the commissioning *Ubuntu 6.06 LTS (Dapper Drake)* was used, which was upgraded to *Ubuntu 8.04 LTS (Hardy Heron)* in the re-commissioning phase in spring 2009.

In order to be able to address more than 4 GByte of memory, the 64-bit (`x86_64` architecture) version was chosen. This is especially important to be able to address large shared memory buffers and to allow the PCI-X based H-RORCs to write their received data into the input buffers of the FEP nodes.

In order to automatize the installation of new nodes, to be able to deploy different packages on different node groups, and to ensure to have the same packages installed on every node group, an installation procedure was defined [108, 109]. Moreover, this procedure has to be able to cope with the heterogeneous hardware of an growing high-performance cluster.

An image of a *golden client* was prepared for every hardware type, which could be rolled out onto the HLT nodes and provided a base installation. A node configuration framework, the *node-config* scripts, has been developed [108], which allows to prepare the different node groups and to make them ready for the application software.

5.4.1.2 Distributed File System

In order to distribute the application software on the HLT cluster nodes, as well as the simulated and recorded, real test data, a distribution mechanism was needed. This mechanism had to fulfill to following criteria:

- Scalability
- Short release and update cycles possible
- High availability
- Existing knowledge inside the HLT group

Several options have been surveyed:

- Sequential copy of application software to the nodes (rsh/ssh based)
- Treelike synchronization (rsync based)
- Distributed file system : NFS[22]
- Distributed file system : AFS[23]

[21] Long Term Support, providing support of security patches and updates for five years
[22] Network File System
[23] Andrew File System

Sequential copy mechanisms and NFS don't scale with a large number of nodes. Treelike synchronization mechanisms needed to be developed at that time, that are able to cope with failures of some branch nodes and still reach every node inside the cluster.

The distributed file system **AFS** [110, 111] was developed to store a large number of small and medium files and to provide read access to them. Its intrinsic design fits perfectly to the needs of the HLT, as the application software is installed once and then only read from the processes of the reconstruction chain.

The stored data or application software is kept on so-called volumes distributed on several file servers. For every volume only one read-write volume (`RW`) exists, but several read-only (`RO`) copies can exist on the file servers. Publishing the stored information from a `RW` volume to `RO` volumes is called "releasing". The AFS clients are installed on every cluster node and use the integrated AFS server load balancing to get their updated copy of a volume as fast as possible. After a read access to the server, the client caches the data locally and does not need any additional network traffic with the server until its cache gets "dirty", meaning the `RO` volume on the server as been altered.

The base paths on the HLT cluster for the `RW` and `RO` volumes are defined as

```
/afs/alihlt.cern.ch/    # for RO volumes
/afs/.alihlt.cern.ch/   # for RW volumes
```

The TAXI interface (as described in section 5.5.1.3) utilizes the AFS release method to update the HCDB from the T-HCDB at the start of the data-taking. The T-HCDB is a `RW` volume, whose content is updated asynchronously with respect to the data-taking. At the start of the run, a snapshot from it is copied to the HCDB `RW` volume, which is then released in order to have a stable read-only copy during the whole run.

5.4.1.3 Backup

In such a large project it is a basic necessity to have a backup system for sensible data, such as user directories and application software under development.

The AFS file system provides backup by design. For the user directories (every user has his own volume) individual backup volumes have been created and have been automatically updated every night. Images of the server infrastructure and application software have been regularly taken and stored the in CERN CASTOR[24] service [112]. The full AFS file system as been backed up irregularly into the CASTOR system as well.

5.4.2 Application Software

The setup of the application software, its distribution to the processing nodes, as well as its provision to users and operators is a major aspect within this thesis. A BASH script based framework as been developed to automatize and standardize the installation process and to organize the application software as described below.

[24]

5.4.2.1 Application Software Packages

In order to have a functional HLT, several application software packages are needed, each serving a different purpose. They are subdivided in 4 topical groups, which are laid out below.

- Data-Transport
- Reconstruction and Analysis
- Interfaces
- RunControl

Data-Transport The *Data-Transport Framework* is the heart of the data processing and serves as a basis for the reconstruction and analysis layer. It has been discussed in detail in section 4.2.2.

Reconstruction and Analysis Four packages assemble the HLT reconstruction and analysis framework.

ROOT The basis of the reconstruction and analysis framework is the C++ based, object oriented data analysis framework *ROOT* [113, 114], which as been developed at CERN. It provides basic and custom containers for data and analysis objects, as well as sophisticated analysis and fitting routines.

Geant3 A software package is needed to describe the detector response of particles to the material. The propagation of particles through the detectors for Monte Carlo simulations is provided by the FORTRAN based *Geant3* [115, 116] package.

AliRoot *AliRoot* (ALIce ROOT) [73, 74] is the ALICE off-line ROOT based framework for simulation, data reconstruction and analysis. It contains all functionalities to calibrate and reconstruct recorded collision data and allows to produce Monte Carlo simulations. A subversion repository [75] keeps not only the development trunk, but also tagged versions. In the HLT both the trunk and the tags are used for development and production respectively.

HLT Analysis Framework The *HLT Analysis Framework* is a part of AliRoot, which can be found in the `$ALICE_ROOT/HLT` folder. It is installed as an extra package, in order to allow for development inside the HLT part of AliRoot, but keeping a fixed AliRoot version. In general, trunk versions of it were used during the commissioning phase in order to enable fast turn around times in the development. All detector reconstruction and analysis components of the HLT are found in this package.

Interfaces The interfaces to the other subsystems such as the *TAXI*, the *PENDOLINO*, and the *HLT–ECS proxy* are discussed in detail in the next section 5.5.1. Their source code in kept in the HLT subversion repository on the HLT cluster.

AliEn *AliEn* [117] is an interface, which allows to connect to the ALICE Grid services from the command line, as well as from AliRoot. It is used in the TAXI interface to access the ALICE OCDB.

RunControl The *RunControl* and the configuration management framework of the HLT are also kept in the HLT subversion repository. Moreover, all configuration objects are stored here as well. It is described in detail in section 5.5.2.

5.4.2.2 Application Software Organization

These software packages have to be organized as such, that they are available on all the processing nodes, but can not be altered by normal users. Therefore, all the software packages have been installed in a separate AFS volumes linked to the common folder `/opt/HLT/<package-name>`. The development and server nodes mount the `RW` volume at this entry point and the processing nodes mount the `RO` volumes.

```
/opt/HLT$
  alien           # Base folder for ALIEN
  aliroot         # Base folder for AliRoot
  analysis        # Base folder for the HLT analysis framework
  control         # Base folder for the RunControl
  data-transport  # Base folder for the data-transport framework
  geant3          # Base folder for Geant3
  installation    # Log files from the installation
  interfaces      # Base folder for all interfaces
  modules         # Base folder for all modules
  root            # Base folder for ROOT
  tools           # Tools for the daily work
```

Two fundamental requirements for the organization of the application software have to be fulfilled.

During the commissioning it is sometimes necessary to fall back to the last version of a software package, which was known to be working. So several versions of software packages have to be managed at the same time.

Furthermore, the future of the HLT cluster hardware and software has to be kept in perspective. The HLT cluster has been a homogeneous set of nodes in the start-up and commissioning phase, but will inevitably diverge in an heterogeneous system in terms of hardware, as well as operating systems.

In order to meet these needs the *modules environment* [118–120] was introduced, for the easy and dynamic modification of the user environment. It allows to load and unload different versions of the application packages and is able to cope with an underlying heterogeneous software environment. Every installed version of a software package is represented by a module, which contains all necessary information to set the proper user environment for this package, so that it can be used.

For simplicity reasons, a top-level module, the **HLT module** has been created, which represents a coherent installation, taking into account all dependencies, of all software packages. It allows to load or unload a complete environment to run the HLT.

For development purposes the users can also have their own installation and module of the packages of the ANALYSIS module (containing a installation of the HLT analysis framework). A simple mechanism to unload the global version of a package and load a private version is provided.

All software packages have been compiled in a production version (identified by the suffix `-prod`) and in a development version (identified by the suffix: `-debug`), allowing for code optimization in the first case. In order to assist with code debugging during the development, the debug information/symbols have been enabled and the compile optimization has been turned off.

```
HLT/v4-19-Rev-01-debug
HLT/v4-19-Rev-01-prod
```

5.5 Configuration

During normal physics data-taking, thousands of processes are needed to reconstruct events on several hundred nodes. All of them have to be started synchronously, orchestrated and kept up-to-date with status information and detector conditions. They communicate continuously with each other and exchange data blocks, containing physics and meta data. The configuration of these processes and their interplay with the interfaces to the outside of the HLT is described in this section.

5.5.1 Interfaces

The interfaces to the other on-line and off-line systems play a key role not only for the data processing itself, but also for the configuration process. They can be divided in three classes according to the handled data types : physics data, configuration data and detector conditions data. A schematic overview of the different interfaces is laid out in Figure 5.15.

5.5.1.1 Physics Data

As discussed previously in section 4.2.1 and section 5.3.2, physics data arrives in the HLT via optical fibers, as well as the HLT trigger decision and HLT payload is sent out via optical fibers. All those incoming and outgoing fibers have been connected on the HLT end to FEP nodes. However, all but the ones for the TOF detector have been connected to LDCs on the DAQ side. Consequently, all but the the TOF DDL links have been tested and commissioned.

ALICE has implemented a common data-format for raw physics data, using a general header, the so-called CDH[25], followed by individual detector payload [65].

A unique event identifier is needed to ensure a proper event building on the DAQ side, as well as the matching of all event fragments in the HLT processing chain. This is achieved by the monotonous increasing EventID, a 64-bit number consisting of the bunch crossing counter (lowest 12 bit), the orbit counter (24 bit), and the period counter (highest 28 bit). The lowest 36 bit are contained in the CDH and identify an event

[25] Common Data Header

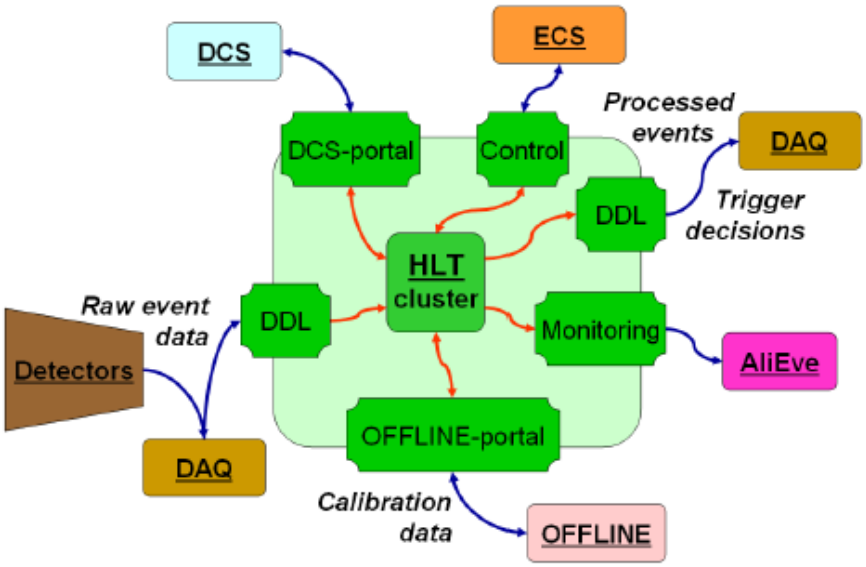

Figure 5.15: Schematic overview of the HLT interface to the other on-line and off-line systems. From [121].

uniquely within 24 minutes[26]. An additional complication is added by the fact that events processed by the HLT can arrive out of sequence, so a period counter was introduced at the level of DAQ and HLT to uniquely identify a given orbit.

The HLT output is particular challenging for the DAQ, as the HLT data is sent only via one out of 10 DDLs per event. However, only half of the outgoing links could be used during the commissioning phase and the first pp collisions. On the DAQ side, two of them have been connected to one LDC each and the DAQ read-out software version has been unable to handle out of sequence arrival of HLT decisions on two links in one LDC.

An actual HLT output data block consists of a CDH, the HLT trigger decision, in the form of a DDL-wise read-out list, and the HLT payload, as described in section 4.2.3 and section 5.6.1 (see also [72] for details). The HLT payload itself is in the form of the HOMER format, the standard HLT access format [122], described in section 5.6.2.2.

5.5.1.2 Configuration Data

The ALICE configuration is setup via the ACT, prior to data-taking. Here, the relevant parameters for the HLT configuration are the DAQ run mode and the identifier string for the HLT trigger menu to be used in the data-taking. These parameters, together with detector and run dependent settings, are communicated from the ECS to the HLT via the HLT–ECS proxy.

ECS Interface The top steering entity in the ALICE environment is the ECS. It is realized as a state machine, which has implemented several stable and transition

[26] The bunch crossing counter identifies the crossing of two LHC bunches within one orbit of 88 μs. With a 24-bit orbit counter, an event can be uniquely identified within $2^{24} \times 88\,\mu s \approx 1476\,s \approx 24$ minutes [72].

states between "OFF" and "RUNNING". The *HLT–ECS proxy* is the interface of the HLT to this state machine (more details can be found in [121]), where the proxy receives the transition commands from the ECS and translates them into commands for the HLT RunManager. Figure 5.16 depicts a schematic overview of the HLT–ECS communication. The details of the different states of HLT–ECS proxy and their translation to RunManager states are described in detail in appendix F.

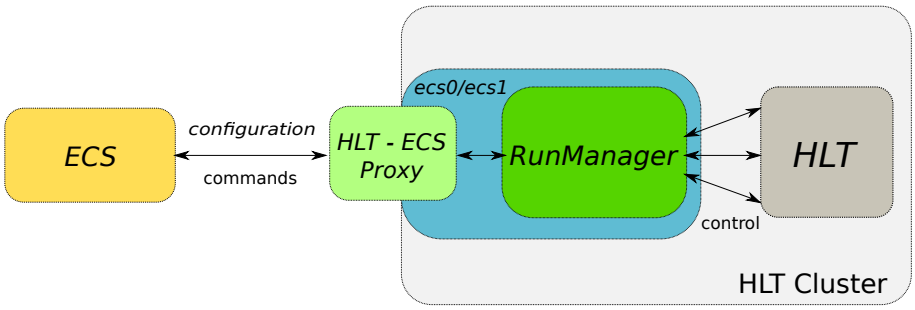

Figure 5.16: Schematic overview of the HLT–ECS communication.

Two sets of configuration parameters are transmitted from ECS to HLT with the *CONFIGURE* and *ENGAGE* commands.

Configure Parameters The *CONFIGURE* parameters, listed below, are needed for the basic setup of the HLT. These parameters contain detector as well as general information, which can be valid over several runs. In such case, the HLT can stay in the "CONFIGURED" state, allowing for a faster start-up of the data-taking.

DETECTOR_LIST The list of active detectors.

BEAM_TYPE The type of the beam, e. g. pp, AA, pA.

DATA_FORMAT_VERSION The expected data format version of the HLT output.

HLT_TRIGGER_CODE The identifier string of the HLT trigger menu, internally in the HLT used as HLT TriggerMenuIdentifier.

HLT_IN_DDL_LIST A list of all incoming DDLs, which are enabled on the DAQ side.

HLT_OUT_DDL_LIST A list of all outgoing DDLs, which are enabled on the DAQ side.

RUN_TYPE The identifier for different run conditions, e. g. PHYSICS, TECHNICAL, COSMICS.

Engage Parameters During the *ENGAGE* command, the run depended parameters are sent, as listed below.

HLT_MODE The running mode of the DAQ, as described in section 3.2.1.2 : A, B, C, D, and E, where D and E are special testing cases of Mode B[27].

RUN_NUMBER The run number of the current run.

CTP_TRIGGER_CLASS The active CTP trigger classes for the current run, associating the trigger bits in the CDH with the actual trigger classes[28].

In order to be able to perform internal tests and benchmarks, a stand-alone version of the HLT part of the ECS state machine was implemented. This allows to configure and run the HLT independently from the ALICE running conditions. A GUI for the proxy, the so-called *ECSGUI*, has been deployed to monitor the HLT state during normal data-taking, but to steer the state machine in stand-alone running (More details can found in [121].). In this case the standard *CONFIGURE* parameters are taken from a configuration text-file in the RunControl environment (`detector/<PARTITION>/setup/ecsproperties.txt`, for details on the folder hierarchy see section 5.5.2), but can be overwritten by the operator. However, the run depended *ENGAGE* parameters have to be provided in any case with the "ENGAGE" command.

5.5.1.3 Detector Conditions Data

Conditions data are needed not only as an input for the detector calibration and, therefore, improving the quality during the reconstruction process, but also to initially configure the HLT reconstruction components. As described before in section 4.3.3, slowly and quickly varying parameters exist and are retrieved from the *Off-line Conditions DataBase* (OCDB) and the DCS database and stored in the *HLT Conditions DataBase* (HCDB). The HCDB represents a subset of the OCDB as not all parameters are needed nor available on-line.

OCDB Interface The HLT is both retrieving parameters from the OCDB and providing calibration data to it. Details of the interplay with the OCDB can be found in [121].

Special components have been implemented in the HLT, which provide detector calibration data on basis of the reconstructed events. All these components inherit from a base class (`AliHLTCalibrationProcessor`), which has been implemented within the context of this thesis and creates calibration data blocks. They are sent to a component the `FXSDumpSubscriber`, running on the vobox0/vobox1 portal nodes, which writes the calibration blocks to a folder based and mysql database supported *File eXchange Server* (FXS). From here, the off-line *Shuttle* process [123] picks up the calibration data blocks after the end of the run. Afterwards, detector depended processes, the so-called *PreProcessors*, compute the actual condition parameters and insert them into the OCDB.

[27] **Mode D** (called B/Test1 in the DAQ environment), allows for data input in the HLT, without any flow control at the DAQ, and no expected output at the DAQ side.
Mode E (called B/Test2 in the DAQ environment), allows for data input in the HLT, with enabled flow control at the DAQ, and no expected output at the DAQ side.

[28] The position of the trigger bits in the CDH can, by design of the CTP, not be predicted and are only known after the CTP configuration. Only then, they are sent to the ECS and consequently to the HLT.

Retrieving information from the OCDB is done via the *TAXI* interface, which is a process on the vobox0/vobox1 portal nodes, running asynchronous to the run structure of the data-taking. Every 30 minutes, it copies a set of predefined subset of OCDB values, or their updates into the TAXI-HCDB (T-HCDB). Only at the start of a run, the HCDB is updated from the T-HCDB in order to allow for stable conditions during the data-taking.

DCS Interface Parameters, which can vary substantially during time of one run, are classified as quickly varying and are retrieved from the DCS database by the *PENDOLINO* interface. Three versions of the PENDOLINO exist, differentiated only by their retrieval intervals of 1 minute, 5 minutes, and 10 minutes. They are started at the beginning of the data-taking and stopped immediately after.

Similar to the off-line Shuttle case, the retrieved data has be to processed, in order to be used during the reconstruction. During the off-line reconstruction, the calibration values are read from the OCDB entries on the basis of the actual time-stamp of the event. However, in the on-line case, the entries do not exist yet, as the event is already processed milliseconds after the read-out and the retrieved parameters are at least one minute old. For this reason, the processing of the retrieved DCS values is similar as performed be the off-line PreProcessor, but includes a prediction step, which is done in the so-called *PredictionProcessor*.

Afterwards, the processed calibration objects are written into the HCDB and continuously updated during the data-taking. Following their creation, the reconstruction components have to be notified that a new calibration object is available. For this reason, a special event, a *DCS Update Event* (`kAliHLTDataTypeUpdtDCS`), is inserted at the beginning of the reconstruction chain and passed through all components. In order to ensure, that all components update their calibration parameters at the same time, these events contain a prospective EventID, after which the new version has to be used.

5.5.2 RunControl

A *Beowulf* cluster includes a head or steering node as a part of its setup. In the HLT this position is taken by the portal ecs0/ecs1 nodes. The *RunControl* is the orchestrating and steering framework of the HLT reconstruction and analysis chain, and was mainly developed within the context of this thesis. As discussed in the previous section, the HLT processes with the RunManager as the top level instance of the RunControl, are steered via the HLT–ECS proxy, directly from the ECS or in stand-alone mode from an operator via the ECSGUI.

The RunControl is executed in the context of a special UNIX user, a so-called *operator* user. Several of those operator users have been created. Every operator implements an own running environment, a so-called *partition*, so that several operators can be in use in parallel at the same time. This practically means several instances of the HLT can be run in parallel. For a detailed description see appendix E.2. Within this thesis *operator* and *partition* are used synonymously.

5.5.2.1 RunControl and TaskManager

One operator steering the HLT reconstruction chain, means controlling ten thousands of jobs on several hundreds of nodes. This is achieved by the TaskManager hierarchy [124].

TaskManager A *TaskManager* is a control process, which steers processes or other TaskManagers. It implements a state machine, whose states are discussed in detail in appendix F.2. A TaskManager can start, stop, configure, connect, or even kill its controlled processes, as well as other TaskManager instances.

A large reconstruction chain is controlled by three levels of TaskManagers. The top level is represented by the so-called *MasterTaskManager*. This process is deployed on the ecs0/ecs1 head nodes and orchestrates a group of *ServantTaskManagers*, which are distributed over the cluster. Each of them is responsible for a logical sub group of the reconstruction chain, like a sub-detector or another global entity. One Servant controls several *SlaveTaskManagers*, where one of them steers all processes of a logical sub groups on one node.

RunManager The *RunManager*, also a special form of a TaskManager, is more than just a connection between the HLT–ECS proxy and the MasterTaskManager. As the highest instance in the HLT RunControl, the RunManager creates the configuration for the TaskManager hierarchy with the help of the received configuration parameters. Furthermore, it steers and controls the whole start-of-run and end-of-run sequences.

TaskManager Configuration An HLT configuration is a list of TaskManagers and processes to be run, their arguments, combined with the information to which processes they have to connect. Additionally, the TaskManager need to know which processes they control and what should be done in which state. This realized with XML[29] configuration files, one for each TaskManager, using the `<SimpleChainConfig>` notation [124].

The configuring of the HLT can be logically divided into two parts:

Physics Environment The actual selection of the reconstruction components, their hierarchy and their dependencies are defined by the physics requirements. This is completely determined by the HLT TriggerMenuIdentifier and is discussed in detail in the next section 5.5.2.2.

Data Flow Environment This part includes the steering part of the chain like the TaskManagers, as well as all the necessary data flow components, which however, are dependent on the physics environment.

The configuration of the data flow environment does not only use the received configuration parameters from the ECS, but also the state of the HLT has to be considered as well. In the current implementation, this is represented as a set of XML configuration files, which are laid out in detail in appendix G.

`globalddllist.xml` List and mapping of ALICE DDL link identifier to the connected FEP node, the PCI address of the corresponding H-RORC, and the DIU.

[29] Extensible Markup Language

`globalsiteconfig.xml` Global site settings for the TaskManager hierarchy and the attached data flow and reconstruction processes, like logging verbosity, data flow settings and shared memory configurations.

`<partition>/siteconfig.xml` Site settings, which can be individual for every partition. Settings here overwrite settings in the `globalsiteconfig.xml`.

`<partition>/nodeconfig.xml` A list of the all nodes associated to one partition and a description of their environment.

5.5.2.2 Component Configuration

The requirements of the HLT output during data-taking and therefore, its configuration, is completely determined by the HLT TriggerMenuIdentifier, a configuration parameter received from the ECS as HLT_TRIGGER_CODE. In order to create an configuration for the physics environment out of this, the possible HLT components, their settings, as well as their dependencies have to be known. Every component as such is a modular object, but together with their dependencies of other components a treelike hierarchy can be spanned.

Additionally, the possibility to add extra components, like special components for calibration, monitoring, or data providing, is needed. In order to keep this collection of possible components as simple as possible for the operator to verify and to maintain, a special, modular configuration schema has been developed and implemented within the context of this thesis.

This configuration schema consists XML based configuration objects, using the `<SimpleChainConfig2>` (SCC2), an enhancement of the original `SimpleChainConfig` (SCC) notation. Details of the notation of SCC and SCC2 are described in [124]. Sample configurations, which were used during the first pp collisions can be found in appendix G.

General Schema The collection of possible configuration objects has 4 logical parts, all kept in as sub folders of `control/hlt_configuration/` in the *control repository*[30].

- Triggermenus : `control/hlt_configuration/triggermenus/`
- Components : `control/hlt_configuration/components/`
- Detectors : `control/hlt_configuration/detectors/`
- Partitions : `control/hlt_configuration/partitions/`

In order to build a configuration, the RunManager, collects all needed and requested configuration objects, which build physics environment. Together with data flow environment, the RunManager creates the XML configuration files for the TaskManagers.

[30] A subversion software repository, which allows versioning and managing of software revisions, used for the configuration of the HLT, in order to be able to track changes and to be able to fall back to a previous working version.

Triggermenus The collection of the configuration objects starts with the trigger menu. In the `triggermenus` folder, a list of possible HLT trigger menus is provided, where each is described by a separate file. The HLT TriggerMenuIdentifier string, received from the ECS as HLT_TRIGGER_CODE, defines the file `<HLT_TRIGGER_CODE>.txt` as the basis of the actual configuration, e. g. :

```
control/hlt_configuration/triggermenus/
   HM-COSMICS-V0001.txt
   HM-LOW_MULTIPLICTY-V0010.txt
   ...
```

The naming convention of the HLT TriggerMenuIdentifier is described in appendix A.1. Every line in one of those files describes the ComponentID of one trigger component, e. g. , for the `HM-LOW_MULTIPLICTY-V0010.txt`:

```
control/hlt_configuration/triggermenus/HM-LOW_MULTIPLICTY-V0010.txt
   BarrelMultiplicity
   BarrelPt_v01
   EmcalClusterEnergyTrigger
   ...
```

This configuration file only specifies the input trigger components. The actual configuration of the global trigger is stored as an HCDB object as described in section 4.4.1 and appendix A.1.

Components Evaluating of the trigger components is the next step. The `components` folder has one sub folder for all the trigger components and one for every detector (SPD, SSD, SDD, TPC, ...). Each sub folder contains XML object files of component configurations, following the structure `hlt_configuration/components/<DETECTOR>/<ComponentID>.xml`, e. g. :

```
control/hlt_configuration/components/
   TPC/CF.xml
   TPC/TR.xml
   trigger/BarrelMultiplicity.xml
   trigger/BarrelPt_v01.xml
   ...
```

The content of those XML files must contain valid `<SimpleChainConfig2>` notation, specifying the full hierarchy of a given component. For instance, the object file for the TPC tracking component in `components/TPC/TR.xml` is shown as Listing 5.1: The component with the ID TR (TRacker) is in `<ALICE>`, belongs to the `<TPC>` detector, works on `<Slice>` level[31] and has a `<Multiplicity>` of 1 (= one tracking process per TPC sector), and will load the TPC CA tracker AliRoot component.

[31] Old HLT internal synonym for a TPC sector, kept for backward compatibility.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2 ID="ITS" verbosity="0x78">
   <infoblock>
      <author>Jochen</author>
      <date>Thu Oct 01 05:15:25 CEST 2009</date>
      <description>Default configuration for TPC - tracker</description>
   </infoblock>

   <ALICE>
      <TPC>
         <Slice>
            <Component ID="TR">
               <ComponentID>TPCCATracker</ComponentID>
               <Options>-neighboursSearchArea 20</Options>
               <Parent>CF</Parent>
               <Shm blocksize="500k" blockcount="1000"/>
               <Multiplicity>1</Multiplicity>
               <Library>libAliHLTTPC.so</Library>
            </Component>
         </Slice>
      </TPC>
   </ALICE>
</SimpleChainConfig2>
```

Listing 5.1: `control/hlt_configuration/components/TPC/TR.xml` An example configuration object file for the TPC tracking component.

Following the dependencies, the `<Parent>` process is a ClusterFinder (CF). Consequently, the file `components/CF/CF.xml` would be read as well. The object list is built up, using *Breadth-First Search* (BFS) to scan through the component folder.

Partitions Configuration objects on the level of a partition, allow to define global settings which are valid for all components in the partition, to add global components like FXS Subscribes, and to define and configure the output to the DAQ.

There is one sub folder for every partition (HLT, TPC, ITS, TRD,...) in the `partitions` folder. Every sub folder contains three more sub folders for different configuration types (production, valid-test, and devel) and the file `allowed_types`. This file contains the names of the configuration types, which will be used to create the object list. The entries for every partition looks like the following:

```
control/hlt_configuration/partitions/
   <PARTITION>/production/
   <PARTITION>/valid-test/
   <PARTITION>/devel/
   <PARTITION>/allowed_types
```

Different configuration types, allow to have several sets of configuration in parallel. While keeping a stable configuration for the production version, a development version

can be edited and used for tests. Each of the different configuration type sub folder can contain configuration XML files in the following schema:

```
control/hlt_configuration/partitions/<PARTITION>/<TYPE>/
   <PARTITION>.xml
   <PARTITION>-<BEAM_TYPE>.xml
   <PARTITION>-<RUN_TYPE>.xml
   <PARTITION>-<BEAM_TYPE>-<RUN_TYPE>.xml
```

Specifying configuration objects on the basis of beam type, run type or even a combination of them, allows to provide ready-to-use configurations for different running scenarios, e. g. pp-PHYSICS or PbPb-TECHNICAL.

Detectors The fourth step is the collecting of detector dependent configurations files. There is one sub folder for every detector (SPD, SSD, SDD, TPC, ...) in the `detectors` folder. Each sub folder can contain a set of default production configurations for the given detector. Similar to the partition folder, also here, four different types can exist, depending on the current beam type and run type. These are searched in the following order

```
control/hlt_configuration/detectors/<DETECTOR>/
   <DETECTOR>.xml
   <DETECTOR>-<BEAM-TYPE>.xml
   <DETECTOR>-<RUN-TYPE>.xml
   <DETECTOR>-<BEAM-TYPE>-<RUN-TYPE>.xml
```

e. g. .

```
control/hlt_configuration/detectors/TPC/
   TPC.xml
   TPC-pp.xml
   TPC-TECHNICAL.xml
   TPC-pp-PHYSICS.xml
```

The content of those XML files must contain a valid `<SimpleChainConfig2>` notation, specifying the full hierarchy of a given component. A sample configuration file for the TPC detector `detectors/TPC/TPC.xml` is shown in Listing 5.2. It enables all DDLs for the TPC and sets special shared memory settings for the front-end buffers on the FEP nodes.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2 ID="TPC" verbosity="0x78">

<ALICE>
   <Sources>
      <DDL>TPC</DDL>
   </Sources>

   <TPC>
      <RORCShm blocksize="200k" blockcount="1000"/>
   </TPC>
</ALICE>
</SimpleChainConfig2>
```

Listing 5.2: `control/hlt_configuration/detectors/TPC/TPC.xml` An example configuration object file for the TPC partition.

Configuration Creation After all configuration objects have been collected, the RunManager combines these objects with configuration parameters from the ECS and the HLT state configuration files. This consolidates all the information for the physics and data flow environment.

The process placement, in order to optimize network throughput and CPU usage is an additional challenging subject. In the current implementation, two possibilities to assign a process to a node exist. A node can be directly assigned in the configuration object. If this is not the case, a simple algorithm distributes the remaining processes to free slots on the nodes. A future enhancement for the configuration process is the use of advanced distribution algorithms.

Together with the information of the process placement on the nodes, the RunManager has enough information to produce the XML configuration files for the TaskManagers. During this process, all needed data flow components are automatically added.

Component Configuration The RunControl configuration is mainly used to determine the necessary components and their hierarchy as well as basic configuration settings of the nodes, the data-transport framework, and the components. Additionally, all AliRoot based analysis and reconstruction components can also be configured HCDB configuration objects, which configure the analysis or reconstructions algorithms themselves. A subsample of these ROOT based calibration objects in the HCDB can be seen below.

```
HCDB/HLT/ConfigHLT/
   BarrelMultiplicityTrigger
   BarrelPt_v01
   BarrelPt_v02
   CosmicsTrigger
   esdLayout
   H_._Barrel_pT_Single_._V0001.001
   H_._Barrel_pT_Single_._V0002.001
```

```
H_._Barrel_pT_Single_._V0003.001
HLTGlobalTrigger
PrimaryVertexFinder
...
```

5.5.2.3 Running Procedure

The start of the data-taking is triggered by the ECS, which brings all detectors and systems from an off or configured state to a running state, such that triggers can be received and events can be recorded. In order to assure a efficient data-taking, several preparations and activities are needed before. All actions after the start of the data-taking are automatic.

Preparation The ALICE physics program, together with the data-taking conditions and the trigger setup describe the ALICE running scenario, which are defined by the ALICE physics coordinator, the run coordinator, and the trigger coordinator. As part of the preparation before the data-taking, the HLT experts on duty have to arrange that the proper configuration files for the components, as well as for the trigger configuration are in place, in order to fulfill the running scenario.

Afterwards, the experts have to ensure, that the correct DLL splitters on the D-RORCs have been enabled and that the ACT is configured with the predefined DAQ mode, as well as HLT_TRIGGER_CODE.

Now, the steering applications have to be launched, beginning with the *RunManager*. During the startup of the *HLT–ECS proxy* the HLT running partition is connected to a DAQ partition. The name of the DAQ partition is defined by the shift-leader and is supplied as a command-line argument to the HLT–ECS proxy. Also the *ECSGUI* can be started now. It is attached to the same partition as the HLT–ECS proxy and enables the experts to watch the state-transitions of the whole system.

Several processes are always running, such as the run-asynchronous TAXI interface. Also the logging and monitoring applications are always active and shall be described here briefly:

InfoBrowser The *InfoBrowser* [125] is a development of the DAQ group and provides a centralized access to logging information of every attached process. As the HLT didn't have such an important tool at the start of its commissioning phase, the data-transport framework, the *InfoLogger Server* (the back-end to the InfoBrowser), as well as the InfoBrowser itself had been adopted [109] to fulfill the needs of the HLT. A screenshot of the InfoBrowser during the data-taking can be seen in Figure 5.17.

SysMES/SysMES GUI The *SysMES GUI* is a front-end to the SysMES cluster management and monitoring application. It represents a graphical overview of the HLT cluster. The health status of every cluster node is depicted by green (everything ok), orange (minor problem), red (major run-critical problem), or black (power off) boxes. Figure 5.18 shows a typical snapshot of the SysMES GUI.

SysMES is able to correct reoccurring issues by itself. In case a problem is identified and solved, a *SysMES rule* can be created using that knowledge. This

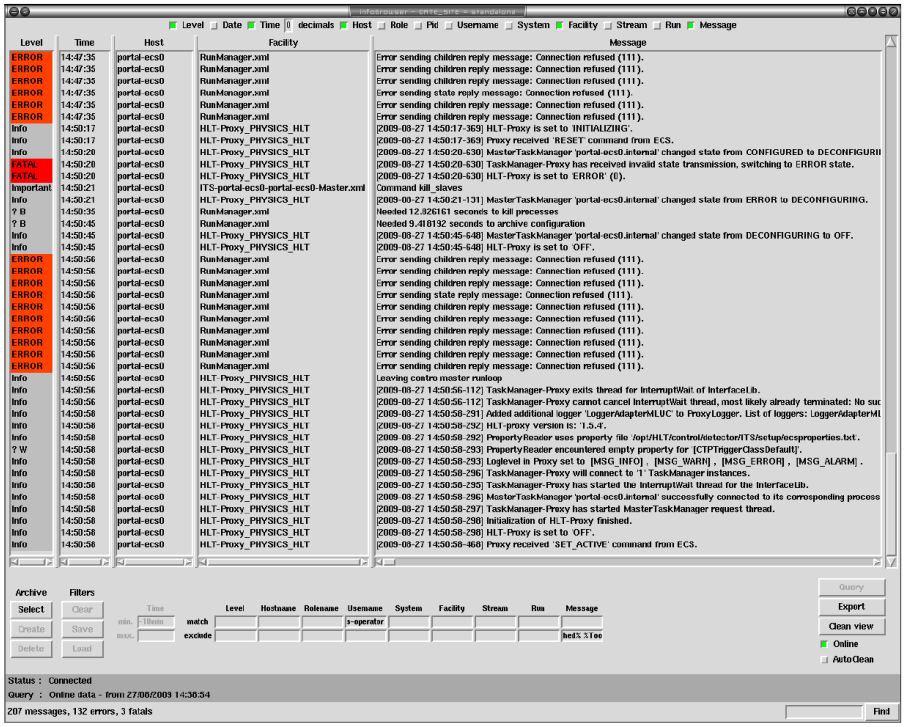

Figure 5.17: A screenshot of the InfoBrowser.

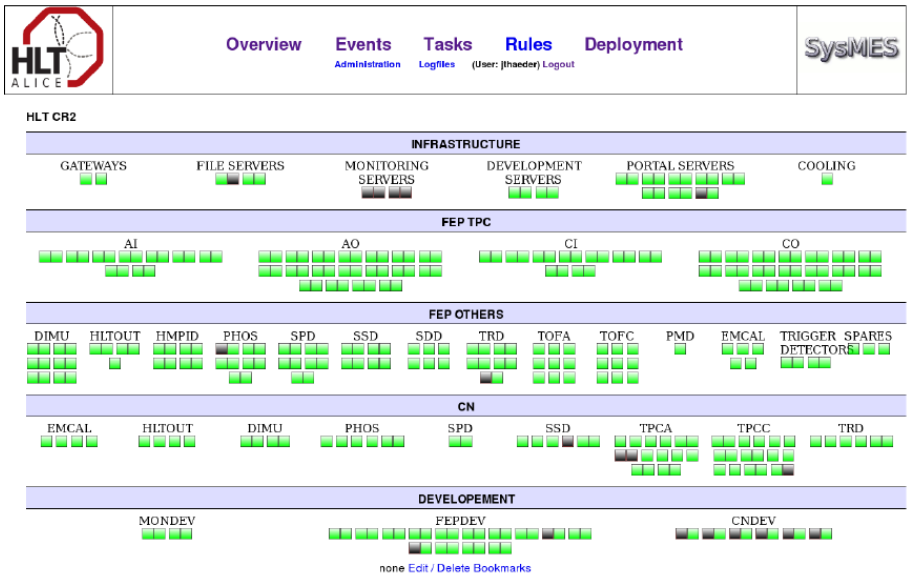

Figure 5.18: A screenshot of the SysMES GUI. Every colored box depicts an entity inside the HLT cluster, where green boxes indicate "everything ok" and black boxes indicate "power off".

rule is able to monitor the symptoms of an issue and can perform automatic actions to resolve a detected error condition. Use cases are e. g. the fill status of a hard disk, which can be automatic cleaned or the failure of a node, which can

be automatically rebooted[32]. The SysMES GUI assists and enables the experts to include new rules and to deploy them onto the cluster. Additionally, it also allows to send simple and advanced tasks (e. g. reboot a single node, shut down or turn on the full cluster) to the HLT cluster nodes and to display their results. Furthermore, an overview of the deployed rules is also given.

Other devices which are monitored by SysMES are the so-called *Cooling-CHARM* and the rack monitoring system. The Cooling-CHARM is an ordinary CHARM card, which is placed below the false floor in CR2 and whose temperature sensors are connected to the water in- and outlets of the rack cooling system. Its status is as well shown on the SysMES GUI. Together with the information from the rack monitoring system, SysMES is able to detected failures in the rack cooling system, as well as electrical power outages. It then is able to turn off affected racks or even the full cluster for cooling issues and to execute a staged, clean shutdown of the infrastructure nodes, connected to an UPS, in case of a electrical power failure.

ESMP2 GUI *ESMP2* is a distributed agent network [126], which provides access to status information of running partitions and their reconstruction chains.

The agents connect to TaskManagers, data flow components, as well as data processing components and even interfaces. This allows them to aggregate all kind of information about the current running reconstruction chain, such as the average processing rate, the number of received events, the number of pending events in the HLT, and the number of running components. Furthermore, also static information like the received HLT–ECS proxy parameters, the loaded environment modules, as well as the active nodes within this partition are collected.

The *ESMP2 GUI* (see also Figure 5.19) is the web based front-end and implements several applications and so-called portlets, which provide simple functionality such as showing statistics trend graphs or status information. Applications are larger software pieces, such as the a browser for logging information or a statistics viewer. It will eventually be the main user interface for HLT RunControl and operations.

Running Procedure The *CONFIGURE* command from the ECS initiates the start-up procedure. First, the HLT configuration is built and distributed, as described before in section 5.5, taking into account the configuration parameters sent by the ECS. Then, the TAXI interface releases the T-HCDB to the HCDB and the PENDOLINOs are started. Now the TaskManagers and all the processes are launched. With the *ENGAGE* command the second phase is initiated in order to bring the HLT into a running state, taking into account the engaging parameters. Now, the data-taking has been started.

At the end of the data-taking the ECS state machine takes care of stopping all processes as described in detail in appendix F. If the configuration parameters are not changing, the HLT can reside in the "CONIFGURED" state, ready to start-up again. Otherwise, it has to be deconfigured first, before it can be configured again.

[32]SysMES is built up of a server and clients, where every node or system management device has its own client, which has a local copy of the possible rules for this device. This enables SysMES clients to react on events even if there is no network connection.

Figure 5.19: A screenshot of the web-based ESMP2. Several portlets are showing the trend-view as well as the current number of monitoring parameters of a running reconstruction chain in the ITS partition. Furthermore, also the received parameters from the HLT–ECS proxy are displayed, as well as an overview of the loaded environment modules and active in users.

Stand-alone Running As already discussed in section 5.5.1.2, the HLT is also able to run in a stand-alone mode, allowing for development tests and benchmarks. The start-up procedure follows the same steps as described above. However, during the launching of the HLT–ECS proxy as well as the ECSGUI the `-standalone` argument has to be appended. The HLT–ECS proxy will now also start a state machine, which mimics the

ECS and allows the ECSGUI to connect to it. Now, the full running procedure can be steered from the ECSGUI, while the configure and engage parameters have to be supplied manually.

In case of normal data-taking, the data source component in the reconstruction chain is the *RORCPublisherComponent*, as described in section 4.2.1, which handles the incoming data from an H-RORC. During the stand-alone running, of course no incoming detector data is available. For this reason two different methods have been developed and deployed to serve data to the reconstruction chain.

- A *FilePublisherComponent* replaces a *RORCPublisherComponent* as data source. This component is able to load a list of so-called DDL-files[33] in to the input buffer of its FEP node and to publish them instead of real incoming data.
- The H-RORCs themselves can also be operated in a replay mode. They are able to buffer a number of DDL-files and can replay them like normal detector data. Using this mechanism, the HLT reconstruction chain can be run in the same manner as during normal reconstruction.

  The H-RORC replay mode has to be enabled and the events have to be loaded onto the H-RORC before the *CONFIGURE* command is issued. After the stand-alone running, the H-RORC replay mode has to be disabled again.

5.6 Data Accessibility

Providing on-line access to reconstructed data was a major objective of the HLT commissioning, which supported not only other detectors during their commissioning phase, but also allowed the HLT itself to study its performance. Furthermore, by operating an on-line event display, the interactions can be visually inspected in real-time. The access possibilities can be divided in a primary data path via the DAQ and secondary data path via *TCPDumpSubscribers* (TDSs) .

5.6.1 Primary Data Path

The primary data path of the HLT leads from the arrival of the raw data at the FEP nodes until the sending of the trigger decision and payload via the HLTOUT nodes to the DAQ. A special output block contains all relevant information from the HLT. This block has a defined format, which is in detailed described in [72] but shall be partly discussed here for completeness. It consists of four parts :

1. **Common Data Header (CDH)** The common header for all DDL raw data blocks [64].
2. **HLT Event Header** The header between HLT and DAQ containing the size and version of the block, as well as the 64-bit EventID, as previously introduced in section 5.5.1.1.
3. **HLT Decision** The HLT Trigger decision.

[33] A DDL-file is a binary file representing an event fragment as normally received from the detector FEE. They can be retrieved from either recorded events or are created during the simulation process. Therefore, every event can be split in 460 single files, one file per DDL link.

4. **HLT Payload** The HLT Payload.

A specialized formatter component, the *HLTOUTFormatterComponent*, collects the all relevant parts and assembles the output block.

In the final usage of the HLT within the ALICE data-taking schema, these HLT output blocks are the main contribution of the HLT. However, already running in the DAQ Mode B, these output blocks are stored and can be used in later off-line analysis for reconstruction and trigger studies.

5.6.1.1 Trigger Decision

Bit 6 in the status field of the CDH, indicates the presence of an HLT decision in that package. As previously described in section 4.4.1, the decision is expressed in a bit-wise read-out list, where every DDL is described by one bit. The read-out list is filled by the *HLT GlobalTriggerComponent*, after bit-wise combining the trigger domains for all contributing triggers of this event with a logical or.

An event is rejected if all bits in the decision are set to zero. However, with the ability to only set its own DDL bit, the HLT can reject all raw data and keep only the HLT payload.

5.6.1.2 Data Payload

In the case that HLT payload is attached to the HLT trigger decision, its presence is indicated by the bit 7 in the status field of the CDH. As the payload itself can consist of many internal data blocks, the individual data blocks are packaged in the HOMER format, which is described in the next section.

It can contain `AliESDEvent` objects, the summary of the HLT reconstruction, but as well every other output block which was created within the reconstruction process. Furthermore, monitoring objects like histograms or data trees can also be included, as well as compressed raw or compressed reconstructed data.

5.6.1.3 Read-Out Trigger

As described before in section 4.2.3, all results from the intermediate processing steps of one event are kept in the shared memory until the event has been completely processed and sent out. The freeing of the space is achieved by the so-called *event-done* mechanism. Here, the last component in the chain, the *HLTOUTSubscriberComponent*, sends a special signal back up the hierarchy, after an event has been successfully transmitted to the DAQ. This signal follows all fan-outs and fan-ins of the treelike reconstruction chain, until it arrives at the very first components, the *RORCPublisherComponents*. On its way, it notifies every component to release all data blocks associated with this event.

An HLT trigger domain does not only contain the list of DDLs to be read-out by the DAQ, but also a list of internal data blocks, which should be included in the HLT payload. The second to last component, the *HLTOUTFormatterComponent*, has also the ability to use the *event-done* mechanism. However, it sends a different signal, the *Read-Out Trigger*, back through the chain, which contains the internal read-out list. Only now all data blocks to be included into the HLT payload are sent to the *HLTOUTFormatterComponent*, avoiding unnecessary network traffic during the previous reconstruction steps.

5.6.2 Secondary Data Path

In addition to the primary data path, there is an additional need for a direct on-line access to reconstructed data. However, this secondary data path has not to interfere with the normal data-taking. This is made possible by the *TCPDumpSubscribers*, which where briefly introduced in section 4.2.3. They are special components, which can be attached anywhere in the reconstruction chain and provide access to their input data via a TCP port. In case a client connects to a TCP port of a specific TDS, the data blocks are not automatically sent. Hence, the data blocks are only transmitted after an explicit request of the client.

The TDS always keeps the latest data block of a given event type, which is particularly important for monitoring objects. In general, they are not computed and sent out for every event, but rather on the basis of time intervals or even EventID intervals.

Different needs of developers, operators, detector experts, shift crew, and even general ALICE members define a selection of use cases, which can be broken down to the type of the data blocks. They can be grouped in two different categories as listed below, where one TDS in general provides only data blocks of one category. However, several TDSs per category can be attached in one reconstruction chain.

Synchronous Data Blocks These are data blocks which are synchronous in respect of the EventID, therefore, belonging all to the same event. Only data blocks arriving in one TDS can be synchronous to each other. A second TDS is attached at a different position in the reconstruction chain and can naturally not be in sync with the first one. An additional requirement is, that all data blocks have to be produced for every event.

Collecting for instance all 216 TPC cluster data blocks of one event is a normal use case. This is especially important in event displays, where all different reconstructed objects naturally have to belong to the same event. Moreover, detector experts or operators can retrieve an event as snapshot of the reconstruction, or even all the raw data of one event for detailed studies already during data-taking.

Asynchronous Data Blocks They are rather used for monitoring objects, like histograms or data trees, which are not regularly updated every event, but are valid for a range of events. Naturally, monitoring objects show the data quality and the performance of the data-taking and reconstruction and make still perfect sense even when updated at different times.

In a normal reconstruction chain there is always one synchronous TDS, collecting the necessary information for the event display and at least one asynchronous TDS for the monitoring objects.

5.6.2.1 Monitoring Trigger

In case of a synchronous TDS for an event display, in general all relevant data blocks for every event are sent to the TDS, which stores them until the next event arrives. This creates unnecessary network traffic, as normally not more than one event every six seconds can be visually inspected. Furthermore, interesting or abnormal events should be selected as well. Therefore, a *Monitoring Trigger* was introduced, which is capable to select those events.

Similar to the *HLTOUT Formatter Component*, the monitoring trigger can utilize the *event-done* mechanism. Therefore, it can sent a special signal through the reconstruction chain, collecting all input data blocks for the TDS. In this case only relevant events arrive at the TDS, which in return reduces the overall network traffic.

5.6.2.2 HOMER Interface

Similar to HLT payload in the primary data path, also all data blocks in the TDS are packaged in the HOMER[34] format [122], establishing a single data format for all outgoing data.

A client, which wants to use this data blocks, first needs to establish a TCP connection to a specific TDS. A TDS is identified by the hostname of the node on which the process is located and the TCP port on which the process is listening. Then, the TDS sends on request a data package and its size, which is stored in a single buffer on the client side. Afterwards, the package needs to be decoded, in order to retrieve the contained data blocks.

The data package itself consists, of a *Meta Descriptor*, containing information about the type and system dependent parameters, like endianness and structure alignment, a *Data Descriptor* section, and a *Data Block* section. Every data descriptor points to the beginning of a data block in the payload section, which is illustrated in Figure 5.20 for two data blocks [122].

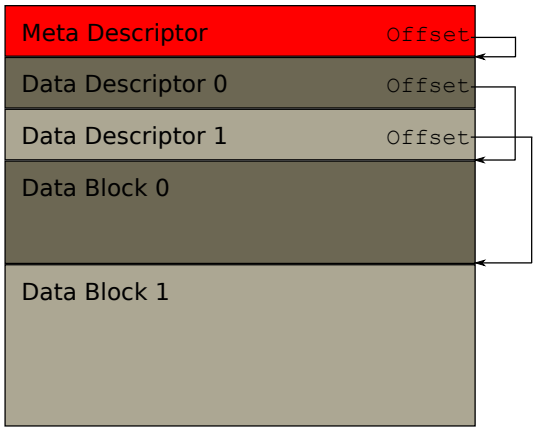

Figure 5.20: The HOMER data format consist of a *Meta Descriptor*, a *Data Descriptor* section, and a *Data Block* section, where the descriptors point to the beginning of the data blocks in the data buffer.

In order to have an interface within AliRoot, allowing to connect to a TDS, to retrieve data packages, and to decode them, the `AliHLTHOMERReader` class exists in the `libAliHLTHOMER.so` library. This allows every AliRoot method to access HLT on-line data, which is schematically shown in Figure 5.21.

[34] HLT Onl-line Monitoring Environment including ROOT

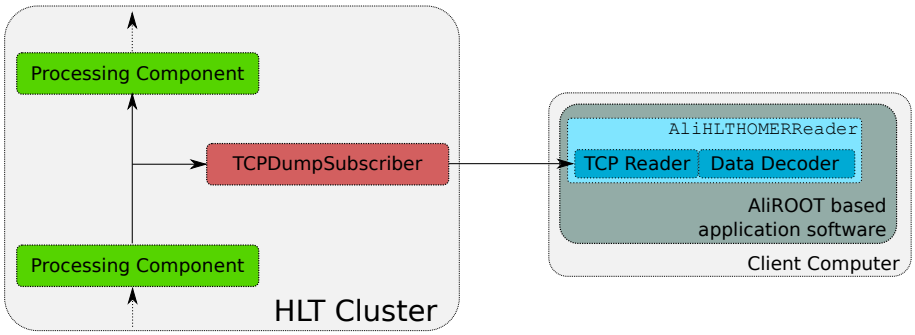

Figure 5.21: A schematic overview of the operation of the HOMER interface. A TCPDumpSubscriber is tabbed into the data stream and provides data blocks to a AliRoot based client. The interface on the client side is implemented in the `AliHLTHOMERReader` class, comprising a TCP reader and data decoder module.

5.6.2.3 HOMER Manager

An HLT on-line reconstruction chain in general includes one TDS providing synchronous data blocks and several, but at least one, TDSs providing asynchronous data blocks. With the automatic placement of the components during the configuration phase the hostnames of the TDSs are not fixed and could change very often. Furthermore, clients from outside the HLT cluster can not access the internal TDSs easily.

Therefore, the *HOMER Manager* was developed and implemented within this thesis, to allow every user in any (ALICE related) CERN network to access the information provided on-line by the HLT within the AliRoot framework. The HOMER Manager forms an abstraction layer, hiding all details of how to find the TDS, to connect to them, as well as the actual data retrieval and decoding. It is implemented in the `AliHLTHOMERManager` class and shall be laid out here in detail. A schematic overview of the HOMER Manager interface can be seen in Figure 5.22.

HOMER Proxy A single point of information about the different TDSs is needed for the clients. This is provided by the *HOMER Proxy*, utilizing the avahi network service[35]. Every TDS announces its name, hostname, and port on startup to the avahi service. After the TDS received the first data blocks, it additionally announces the data types of blocks which can be provided. This information is updated if later on new, unannounced data blocks appear.

The *tcpdump* module of the *clusterapi* package [108, 109], the so-called *HOMER Proxy*, is as well a part of this avahi network and keeps an updated list of all currently active TDSs in the HLT cluster. This list can be retrieved via XML-RPC[36] by contacting the HOMER Proxy on port 19999.

Clients outside the HLT cluster can neither see nor reach the cluster nodes directly. In order to allow the usage of the HLT data, a port forwarding was introduced. It is

[35] A network service for service discovery on a local network via multicast DNS. If a new client connects to the avahi, it provides its service information to all other clients. All clients host all information [127].

[36] XML Remote Procedure Call. A remote procedure call using HTTP as the transport and XML as the encoding [128].

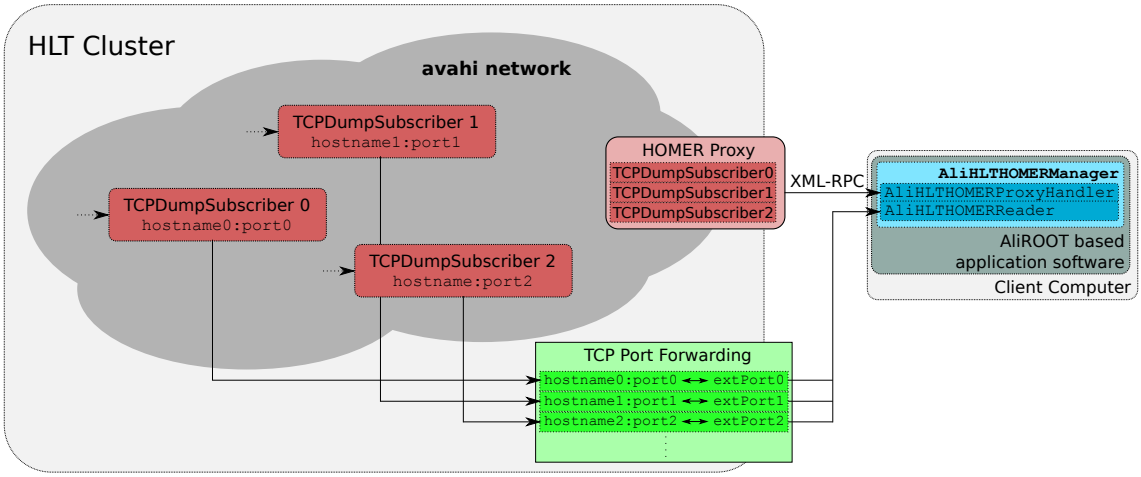

Figure 5.22: A schematic overview of the HOMER Manager interface. The HOMER Proxy keeps an updated list of all TCPDumpSubscribers and provides them via XML-RPC to the HOMER Manager. Afterwards, the client is able to retrieve data packages from the TCPDumpSubscribers via TCP port forwarding.

active on the portal nodes and maps four ports[37] on every cluster node to a unique port on the portal nodes. Details of the mapping are described in appendix B.3. The list of TDSs kept by the HOMER Proxy contains these external ports as well.

Three networks (HLT, DCS/ACR, GPN) are the main use cases for the HOMER interface. Therefore, the portal nodes for the GPN network (vobox0/vobox1), as well as for the DCS/ACR network (dcs0/dcs1) host an HOMER Proxy on both their nodes, as listed in appendix E.3. The avahi network naturally provides them all with the same information, and therefore, they are all synchronized. Clients inside the HLT cluster can use any of those.

HOMER ProxyHandler As a part of the HOMER Manager, the *HOMER ProxyHandler* class was implemented (`AliHLTHOMERProxyHandler`), which takes care of the communication between the HOMER Proxy and the HOMER Manager (`AliHLTHOMERManager`). Its main task is the creation of a list of possible data sources for the HOMER Manager. The full class layout can be seen in appendix H.3.

In the initialization phase, the HOMER ProxyHandler automatically detects in which network the client resides via the method :

```
void IdentifyRealm();
```

Afterwards, the list of possible data sources needs to be created, which is done by calling the method

```
Int_t FillSourceList(TList *srcList);
```

which utilizes the methods

[37] 49152, 49153, 49154, and 49155

```
Int_t RequestXmlRpcResponse();
Int_t ProcessXmlRpcResponse();
```

to create a TCP connection to the HOMER Proxy and to retrieve the list of TDSs via XML-RPC. Afterwards, the received list is processed and the list of sources created. If the client wants to recreate or update the list of sources later on, the same method can be called again. Every source is implemented as class `AliHLTHOMERSourceDesc`, which is described in Listing 5.3. The full class layout can be found in appendix H.1.

```
class AliHLTHOMERSourceDesc : public TNamed {
public:
  // . . . .
private:
  // . . . .

  TString fSourceName;          /** Name of Source */

  // -- Service Specifications --
  TString fHostname;            /** Name of HOMER Node */
  Int_t fPort;                  /** Name of HOMER port */

  // -- Data Specifications --
  TString fDataType;            /** HLT DataType */
  TString fDetector;            /** Detector Name, corresponds to HLT origin */
  ULong_t fSpecification;       /** HLT Specification */
  // . . . .
};
```

Listing 5.3: The relevant members of the class `AliHLTHOMERSourceDesc` (Code formatting adapted to fit in this text).

One TDS in the HLT cluster does not directly correspond to one source object, but it can contribute to several. Moreover, a source object corresponds to a data block type of one TDS. Therefore, a source object is described by a name together with the hostname and port of a TDS, but also contains HLT internal information of the data type, data origin and data specification.

Even if it is currently not used, it is implemented as such, that a client is able, after retrieval of the sources list, to specifically select sources and then in the event loop, only retrieve the corresponding data blocks.

HOMER Manager The `AliHLTHOMERManager` implements the main class of the *HOMER Manager*, whose full class layout can be found in appendix H.4. It provides methods to get the list of sources via the HOMER ProxyHandler and to connect to the TDSs of the selected sources.

```
/** Create Sources List from HOMER-Proxy */
virtual Int_t CreateSourcesList();
/** Connect to HOMER sources, of a certain detector. */
```

```
Int_t ConnectHOMER( TString detector="ALL" );
/** Disconnect from HOMER sources */
void  DisconnectHOMER();
/** Reconnect from HOMER sources */
Int_t ReconnectHOMER( TString detector);
```

After the connections to the TDSs have been established, the HOMER Manager provides the possibility for an event loop.

```
/** Loads the next Event, after being connected */
virtual Int_t NextEvent();
/** Loads the next Cycle, after being connected */
virtual Int_t NextCycle() { return NextEvent(); }
```

For every event the all HOMER packages are decoded and the contained data blocks are sorted in the lists of synchronous and asynchronous data blocks.

```
/** List containing asychronous blocks */
TList* fAsyncBlockList;
/** List containing sychronous blocks */
TList* fBlockList;
```

In case the client wants to return to a previous event, an event buffer has been implemented, which only keeps the last 15 synchronous block lists, as asynchronous ones are not event specific. This allows the client to navigate forward and backward within the event buffer.

HOMER Blocks The heart of the HOMER Manager are the data blocks, each represented by a `AliHLTHOMERBlockDesc` object, which are kept in the block lists (The full class layout can be found in appendix H.2). Each object corresponds to one data block from a HOMER package and contains its name, size, and the retrieved data block itself. Furthermore, the HLT internal fields data type, data origin and data specification are provided.

```
class AliHLTHOMERBlockDesc : public TObject, public AliHLTLogging {
public:
  // . . . .
private:
  // . . . .

  // -- Block properties --
  Char_t* fData;             /** Pointer to data of the block */
  ULong_t fSize;             /** Size of data */
  TString fBlockName;        /** Block Name */
```

```
  // -- Data flags --
  Bool_t fIsTObject;          /** States if block contains a TObject */
  Bool_t fIsRawData;          /** States if block contains a raw data */

  // -- TObject properties --
  AliHLTMessage* fMessage;    /** AliHTMessage object containg a TObject */
  TObject*       fTObject;    /** TObject extracted out of AliHLTMessage */
  TString        fClassName; /** Class Name of the block */

  // -- Data Specifications --
  TString fDataType;          /** HLT DataType */
  TString fDetector;          /** Detector Name */
  ULong_t fSpecification;     /** HLT Specification */
  // . . . .
};
```

Listing 5.4: The relevant members of the class `AliHLTHOMERBlockDesc` (Code formatting adapted to fit in this text).

The output blocks from HLT components can contain raw detector data, reconstructed data in form of simple C structures, but also in form of `TObject` based ROOT objects. A `TObject` can reside at several places in the memory. In order to send these objects, their parts have to be collected and streamed via an `AliHLTMessage` into the shared memory [76], and therefore, also into HOMER data blocks.

An `AliHLTHOMERBlockDesc` object detects if it contains raw data, a simple C structure, or an `TObject`. In the latter case, the `ALiHLTMessage` is unpacked and direct access to the TObject as well as its classname is provided.

5.6.2.4 Applications

The HOMER Manager has several applications and is the main access path to on-line reconstructed data from the HLT.

Detector Commissioning During the detector commissioning, simple AliRoot macros where used by the detector experts to get snapshots of raw data as well as reconstructed data, such as space points. Furthermore, it was used monitor performance histograms during the development phase of reconstruction and calibration algorithms.

Data Quality Monitoring In the *Data Quality Monitoring* (DQM) the overall performance of the read out data is monitored during the data-taking. It is implemented in the AMORE[38] framework, where a client for each detector exists. In general, a client is fed by processed raw data of a small subset of all events.

The HLT client can directly use the outcome of the HLT reconstruction and internal monitoring via the HOMER interface. Therefore, an extra method was added to the HOMER Manager in order to ease the communication with AMORE:

```
virtual Int_t NextCycle();
```

[38] Automatic Monitoring Environment [129]

The HLT information in the DQM was used by the three ITS detectors in the commissioning during the first pp collisions. They used the on-line reconstructed clusters in order to monitor the occupancy of the ITS.

Event Display An on-line event display is an important tool to visually inspect the read out events and to check for anomalies. The standard off-line tool for an event display in ALICE is the OpenGL[39] based *AliEVE*, which can be found in AliRoot under `$ALICE_ROOT/EVE` [131] and is based on the EVE package of ROOT [132].

The HLT uses this standard tool as *AliHLTEve*, which can be found in AliRoot under `$ALICE_ROOT/EVE/EveHLT`. It is a customized user front-end in order to profit of implementations, which already have been done for AliEVE. Therefore, it is an advantage that the HLT uses the same data structure for the event summary object (`AliESDEvent`) as off-line. In the off-line case the data is read from files, where in the on-line case, the data is retrieved via the HOMER interface.

The class `AliEveHOMERManager`, a daughter class of the `AliHLTHOMERManager`, forms the back-end of AliHLTEve. It connects to the HLT cluster and provides the data blocks for the visualization. This allows every ALICE member (with an AliRoot installation) within the CERN network to use the HLT on-line event display. Several examples of 2D and 3D event visualization, as well as displayed histograms can be found in section 5.2.

[39] An industry standard for graphics applications [130]

6. High-p_{t} Trigger

Triggering is the final step in the HLT processing. However, in the first year of data-taking the ALICE detector was used to collect so-called *minimum-bias data*, which only has a minimum of bias through the interaction trigger and no physics triggers are applied. In this data-taking scenario, the HLT could only play a role in limiting the amount of stored data.

During the 2010 pp data-taking period ALICE planned to record around 10^9 minimum-bias interactions, for which the storage space on the permanent storage was reserved. Being still in the commissioning phase as well, the LHC performance increased much faster then expected. The development expressed in number of recorded interactions can be seen in Figure 6.1. In September 2010, the progress of the LHC, as well as the amount of used storage space was reviewed. It was found, that if the luminosity increase of the LHC would progress with the same speed, ALICE's estimated storage space would run out before the end of October, the end of 2010 pp data-taking campaign.

Not a large number of intelligent L0, L1 triggers existed during that period and so the natural way to reduce the amount of stored data was using the HLT. On that account, a trigger on high-momentum particles was implemented, tested and commissioned as part of this thesis and is described in this chapter. As the commissioning of any accelerator or detector, and so also especially of the LHC, is not a very predictable process, in the end the LHC did not progress with the same speed and there was no need to reduce the amount of stored data. However, this trigger is a key prototype implementation for all later trigger implementations, as for the first time a trigger was fully integrated within the HLT global trigger and the data-taking process. In addition to this the trigger might be used in future pp or Pb–Pb data-taking.

6.1 On-line Tracking Performance

It is crucial for a trigger based on on-line reconstructed tracks, to have an understanding of the on-line tracking performance. The reconstruction algorithms for the TPC data running in the HLT on-line environment are different from the ones running in the off-line environment, as it has been laid out before. Due to strict requirements in memory consumption and processing time, optimized algorithms are used in the HLT. It is clear by design, that the on-line algorithms don't provide the same quality of data as off-line algorithms.

Detailed studies have been carried out in order to evaluate the the performance of the on-line cluster finding and tracking algorithms and its comparison to the off-line ones. Details of these studies can be found here [79, 81, 134]. However, some key figures on the tracking performance have been included in this section for completeness.

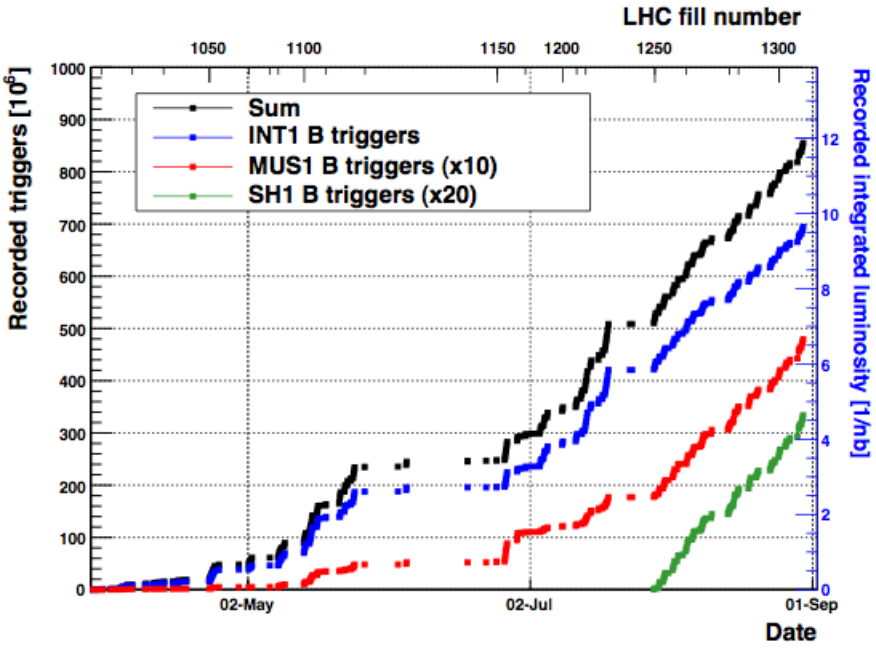

Figure 6.1: Integrated number of events the in 2010 pp data-taking period until beginning of September for the interaction trigger *INT1B* as well as for two rare triggers. From [133].

6.1.1 Monte Carlo Simulations

With the help of Monte Carlo simulations the ideal case can be studied. They are used to evaluate the tracking efficiency and the p_{t} resolution. Furthermore, they are used to judge on general properties of the reconstructed tracks.

This study was performed on simulated minimum bias pp events at $\sqrt{s} = 7\,\mathrm{TeV}$, which have been simulated with the AliRoot framework using the Pythia 6.2 [135] event generator and Geant3 in order to propagate the particles through the detector. In order to mimic real data, the simulation is anchored to detector conditions of already recorded runs[1].

The on-line reconstruction of the HLT is mimicked in the last stage of the simulation process after all the detector response has been created. During the reconstruction, the output of both HLT and off-line reconstruction are written out into one file. This consists of two separate trees of event summary objects of the type `AliESDEvent` for both on-line and off-line reconstructed events.

Using the AliRoot based analysis framework these events can be analyzed. Different analysises are structured in different tasks, based on the class `AliAnalysisTask`. The histograms in this chapter have been created with the standard QA[2] task of the TPC, which can be found within the AliRoot software package (`$ALICE_ROOT/PWGPP/TPC`). It was originally developed to asses the performance of the TPC off-line reconstruction, but has been extended within the scope of thesis to also evaluate the performance of the HLT on-line reconstruction.

[1] For this study the LHC data-taking periods LHC10c and LHC10d have been used, which relate to the Monte Carlo production LHC10f6a (174 M events) and LHC10d4 (72 M events) respectively.

[2] Quality Assurance

The tracking efficiency for all and for findable[3] tracks is comparable for both algorithms as shown in Figure 6.2 and Figure 6.3 respectively.

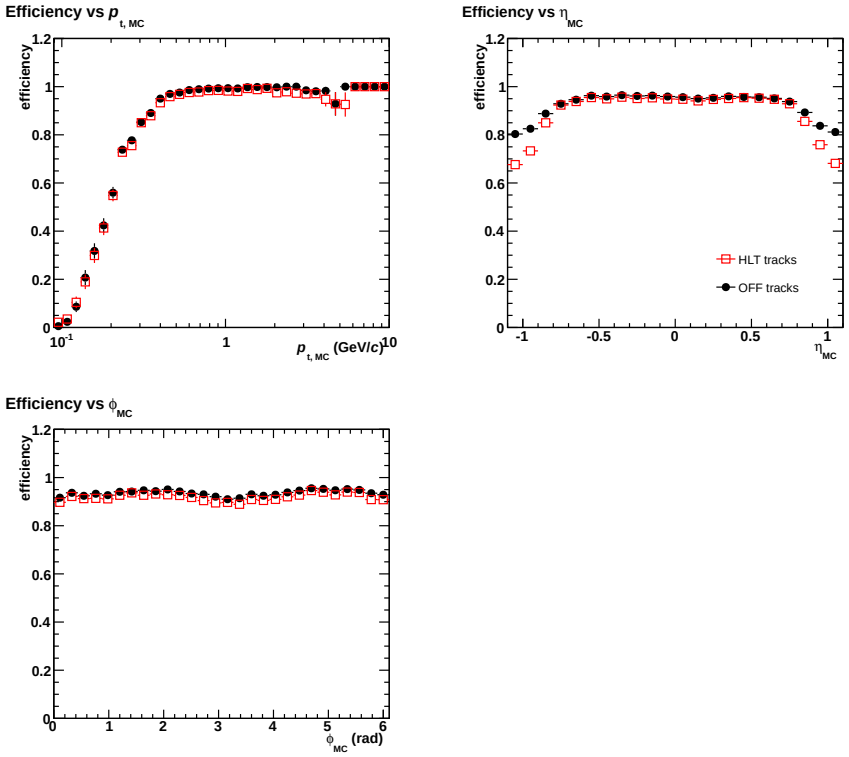

Figure 6.2: The TPC reconstruction efficiency for run 125633 (LHC10f6a) is shown for HLT reconstructed tracks in red and off-line reconstructed tracks in black. The figure shows counter-clockwise from the bottom left, the ϕ_{MC}, $p_{t,MC}$, and η_{MC} dependence. No significant difference between the two is observed within the full tracking acceptance $|\eta| < 0.8$.

Due to missing re-fitting steps in the on-line tracking, the p_t resolution of the on-line tracks is worse by a almost a factor of 1.5 as shown in Figure 6.4. The p_t resolution for off-line reconstructed TPC tracks at $p_t = 7\,\mathrm{GeV}/c$ is 3.8 % and 5.9 % for HLT on-line reconstructed tracks.

The difference of the HLT and off-line cluster finding and tracking algorithms can be also seen in performance histograms as shown in Figure 6.5. Here, the difference in the number of clusters associated to reconstructed tracks accounts for the different tracking algorithms. The general performance is described in the p_t, η, and ϕ distributions. Moreover, the DCAz and DCAr[4] distributions to the primary vertex can give indication on the applied calibration.

In general the HLT tracking algorithm finds more tracks then the off-line tracking algorithm and associates more clusters to the track. Moreover, the worse p_t resolution of HLT tracks can be accounted on these clusters, due to slightly worse cluster resolution in the HLT. The broader DCAz and DCAr distribution account for only one fit of the

[3] A findable track is a particle, which traversed at least 70 cm of the active volume of the TPC and, therefore, can be found by the reconstruction algorithms.

[4] *Distance of Closest Approach* (DCA) in z and r direction

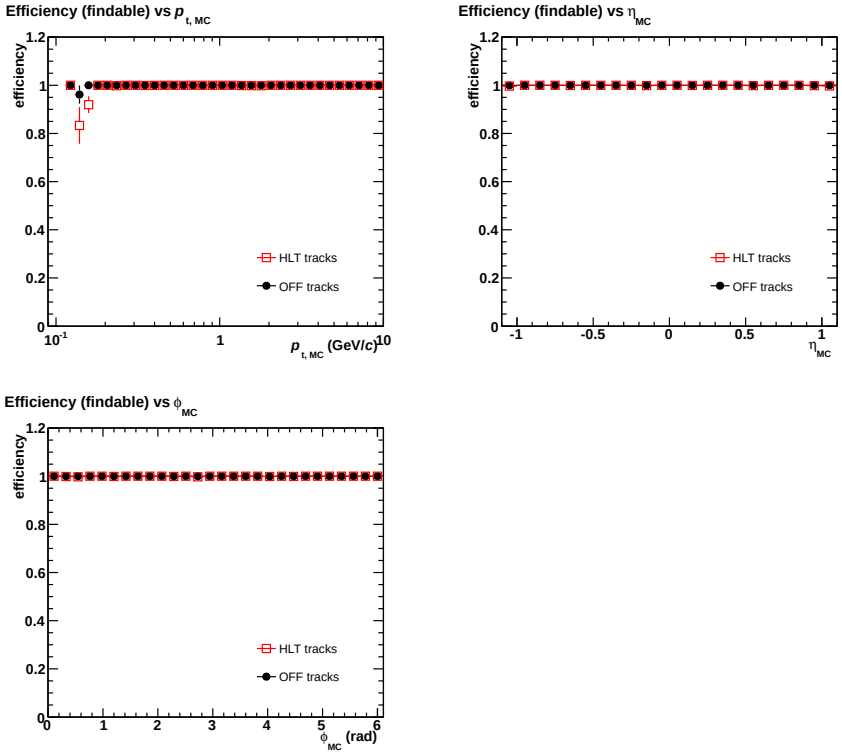

Figure 6.3: The TPC reconstruction efficiency of findable tracks for run 125633 (LHC10f6a) is shown for HLT reconstructed tracks in red and off-line reconstructed tracks in black. The figure shows counter-clockwise from the bottom left, the ϕ_{MC}, $p_{\mathrm{t,MC}}$, and η_{MC} dependence. No significant difference between the two is observed.

track parameters in the HLT in contrast to three of the off-line tracking algorithm.

6.2 Single Particle Trigger

During the commissioning phase, ALICE needed a large number of tracks produced by cosmic particles traversing the active volume of the ITS, in order to align the tracking detectors. Already then, a very simple, geometrical trigger was implemented and used, which only counted the number of TPC tracks, passing the volume of the SDD detector [136, 137].

In this running scenario, the ACORDE, SPD, and TOF based L0 trigger algorithms have been opened up such, that there was only a very loose selection, which resulted in a read-out rate of $\approx 200\,\mathrm{Hz}$. The HLT cosmics trigger accepted only $\approx 16\,\%$ of the read out events, with a very good purity and efficiency. The storage rate was reduced by this to $\approx 30\,\mathrm{Hz}$ [47].

The first HLT trigger algorithm for pp collisions, which was studied here, is based on single high momentum tracks traversing the active volume of the TPC. This section describes its algorithm and the obtained results.

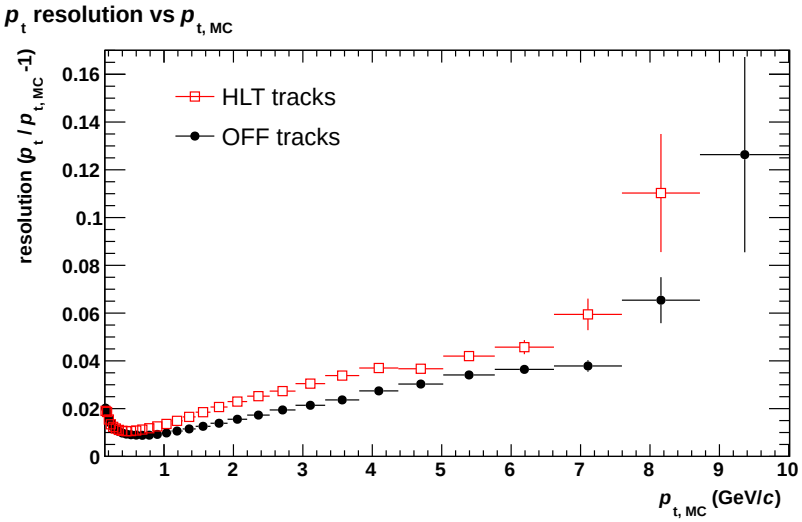

Figure 6.4: The p_{t} resolution TPC tracks for run 125633 (LHC10f6a) is shown for HLT reconstructed tracks in red and off-line reconstructed tracks in black. The p_{t} resolution for HLT reconstructed tracks above a $p_{\mathrm{t}} > 1\,\mathrm{GeV}/c$ is worse then for off-line reconstructed tracks.

6.2.1 Description

In order to enhance the sample of events containing high-p_{t} particles, this trigger algorithm was implemented and tested, keeping in mind the good tracking efficiency but slightly worse p_{t} resolution of the on-line reconstructed tracks. Similar to the on-line tracking performance studies, the Monte Carlo production LHC10f6a (174 M events) and LHC10d4 (72 M events) have been used, which are anchored to the LHC data-taking periods LHC10c and LHC10d respectively.

For this trigger study all events have been considered, which have been compared with those passing the event selection, requiring a valid physics trigger and a reconstructed primary vertex. This was done in order to mimic the conditions of on-line reconstruction in the HLT.

A loose track selection on the trigger tracks has been applied which is listed in Table 6.1. The tracks have been selected to be in the active volume of the TPC and are pointing towards the primary vertex. In order to have some basic criteria of the quality of the tracks, a minimum of 60 TPC clusters associated to each track as been required.

η	$< \lvert 1.0 \rvert$
DCAr	$< 3.0\,\mathrm{cm}$
DCAz	$< 10.0\,\mathrm{cm}$
p_{t}	$> 0.3\,\mathrm{GeV}/c$
N TPC clusters	> 60

Table 6.1: Loose track selection for trigger particles pointing to the primary vertex.

Different p_{t} thresholds for the trigger particle have been investigated. In order to account for the degrading p_{t} resolution for higher p_{t} tracks, the required track length in

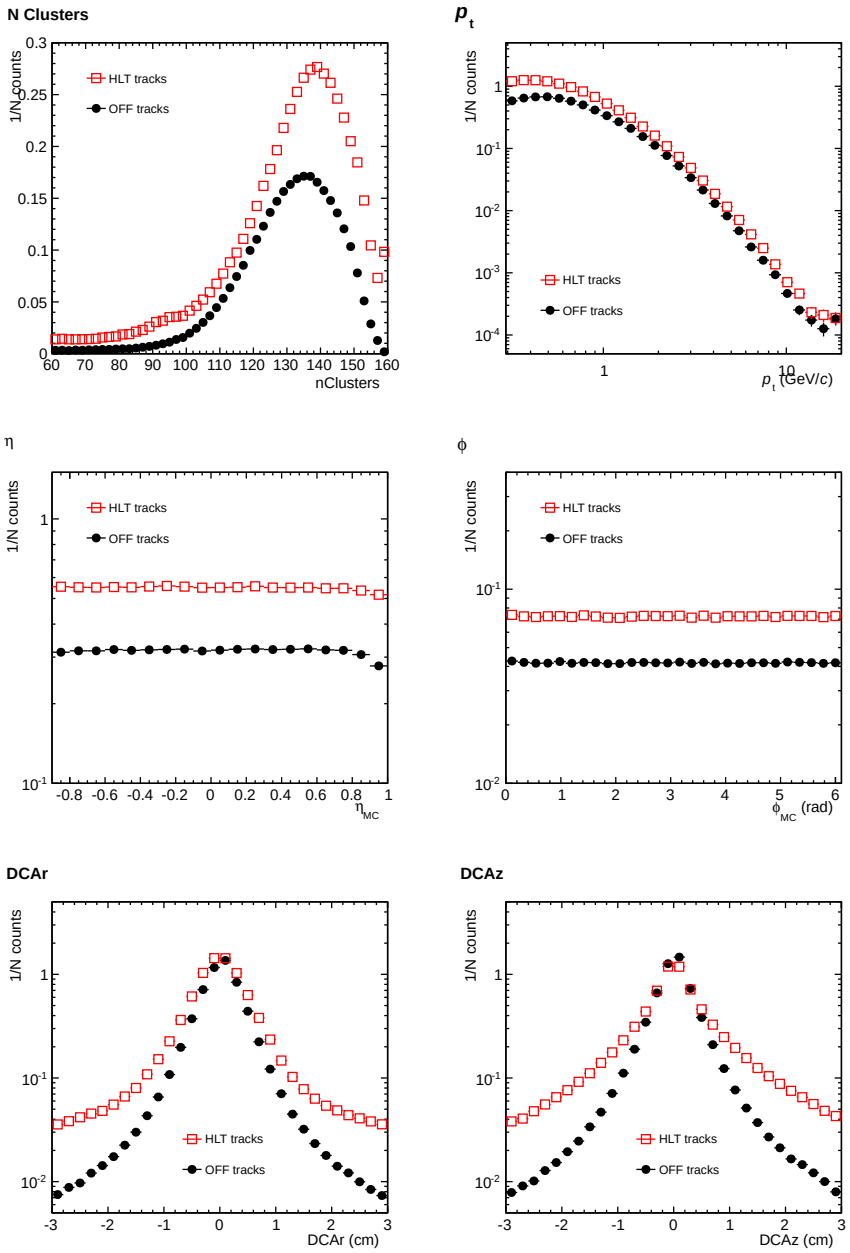

Figure 6.5: General performance histograms for run 125633 (LHC10f6a) are shown for HLT reconstructed tracks in red and off-line reconstructed tracks in black. The figure shows from top down and from left to right, the number of associated clusters, the p_{t}, η, and ϕ distributions, as well as the the DCAz and DCAr distributions.

the TPC was adjusted to the trigger threshold, as longer tracks allow for a better p_{t} resolution than shorter ones. The investigated trigger thresholds and the associated minimum number of TPC clusters are listed in Table 6.2.

p_t **Threshold (GeV/c)**	**Min N TPC Clusters**
1.0	80
2.0	80
2.5	80
3.0	100
5.0	110
7.0	120
10.0	140

Table 6.2: p_t trigger thresholds with associated minimum number of TPC clusters.

6.2.2 Results

In order to allow to compare the triggered results, three different trigger scenarios have been tested as described below. For the efficiency study all triggers have been applied on the off-line reconstructed tracks, as these are the basis for later physics analysis.

- An HLT trigger (HLT) based on HLT reconstructed tracks.
- An off-line trigger (OFF) based on off-line reconstructed tracks.
- A trigger on the Monte Carlo information (MC) based on simulated particles.

An important measure for a trigger algorithm is the trigger efficiency and the trigger purity. in order to be able to retrieve this measure, the number of fake triggered events HLT_{Fake} and missed triggered events HLT_{Miss} have to be defined. HLT_{Fake} events are the events which have been triggered by the HLT, but wouldn't have been triggered by the off-line algorithm.

$$HLT_{Fake} = \ !\, OFF_{Triggered} \ \&\& \ HLT_{Triggered}$$

Opposed to the HLT_{Miss} events, which are the events which would have been triggered by the off-line algorithm, but have not been triggered by the HLT trigger.

$$HLT_{Miss} = \ OFF_{Triggered} \ \&\& \ !\, HLT_{Triggered}$$

Now the trigger purity $Purity_{HLT}$ can be defined as the fraction of the true triggered events ($HLT_{Triggered} - HLT_{Fake}$) over all triggered events.

$$Purity_{HLT} = \frac{HLT_{Triggered} - HLT_{Fake}}{HLT_{Triggered}}$$

The trigger efficiency $Efficiency_{HLT}$ can be defined as the fraction of all off-line available events ($OFF_{Triggered} - HLT_{Miss}$) over all possible triggered events.

$$Efficiency_{HLT} = \frac{OFF_{Triggered} - HLT_{Miss}}{OFF_{Triggered}}$$

A measure of the amount of reduced events is given by the reduction factor $ReductionFactor$, which describes the fraction of all events to the triggered ones. In order to estimate the

reduced data volume, a weighted reduction factor has been introduced as well, taking into account the number of all tracks of each event.

$$ReductionFactor = \frac{Nevents_{Total}}{Nevents_{Triggered}}$$

The trigger efficiencies and trigger purities for different triggers based on different input tracks are applied on different tracks as shown in Figure 6.6 and Figure 6.7 respectively. All figures in the section are based on the LHC10f6a data production anchored to LHC10d data-taking period. A degradation of the trigger efficiency, as well as the trigger purity going to higher p_{t} can be observed, which is due to the worsening momentum resolution of the input tracks. Where trigger efficiency stays always above 80 % the trigger purity drops significantly above a trigger $p_{\mathrm{t}} > 5\,\mathrm{GeV}/c$.

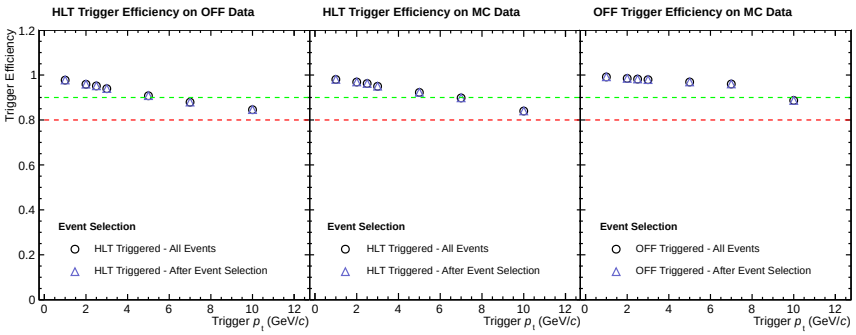

Figure 6.6: The trigger efficiency of triggers based on different input tracks and applied on different tracks, based on the LHC10f6a data set. From left to right, this figure shows the HLT trigger efficiency on off-line reconstructed tracks, the HLT trigger efficiency on MC particles, and the off-line trigger efficiency on MC particles. A slight degradation of the trigger efficiency going to higher p_{t} can be observed, which is due to the worsening momentum resolution. The green and the red horizontal lines indicate a trigger efficiency of 90 % and 80 % respectively.

The effect of an applied trigger on the p_{t} distribution of the off-line reconstructed events has been studied as well, as shown in Figure 6.8. Taking the ratio for different trigger thresholds with the un-triggered distribution an efficiency can be retrieved which is shown in Figure 6.9. They can serve as correction factors in a final physics analysis or define reliable regions of the triggered spectrum. The inefficiencies going to larger p_{t}'s is due to the worsening p_{t} resolution for both the HLT and off-line reconstructed tracks.

The achieved reduction factors can be seen in in Figure 6.10, which shows a steep increase with the trigger p_{t}. A trigger threshold of $p_{\mathrm{t}} > 5\,\mathrm{GeV}/c$ has already a reduction factor of > 100. Moreover, the trigger efficiencies, the trigger purities, and the reduction factors are given in Table 6.3, together with the resulting recording rate, assuming a read-out rate of 800 Hz, limited by the SDD detector.

Taking these results into account and the necessity for the reduction of the data volume by a factor of 4, already a trigger threshold of $p_{\mathrm{t}} = 2\,\mathrm{GeV}/c$ fulfills these needs. The trigger efficiency and the trigger purity are both above 95 % in this range.

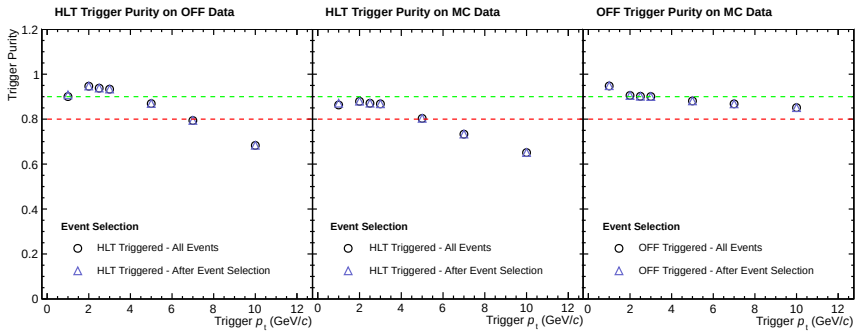

Figure 6.7: The trigger purity of triggers based on different input tracks and applied on different tracks, based on the LHC10f6a data set. From left to right, this figure shows the HLT trigger purity on off-line reconstructed tracks, the HLT trigger purity on MC particles, and the off-line trigger purity on MC particles. A clear degradation of the trigger purity going to higher p_t can be observed, which is due to the worsening momentum resolution. The green and the red horizontal lines indicate a trigger purity of 90 % and 80 % respectively.

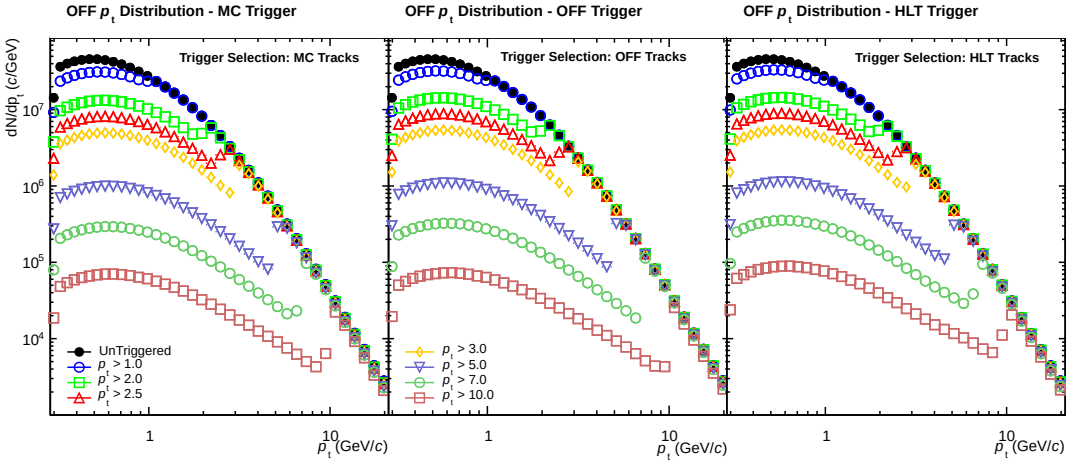

Figure 6.8: The p_t distribution of the off-line reconstructed tracks for different trigger inputs and different trigger thresholds. This figure shows from left to right the p_t distribution of the off-line reconstructed tracks with an applied trigger with of Monte Carlo particles, off-line reconstructed tracks and HLT reconstructed tracks. Different colors indicate different trigger thresholds, where the untriggered case is marked as black circles.

Trigger p_t (GeV/c)	**1.0**	**2.0**	**2.5**	**3.0**	**5.0**	**7.0**	**10.0**
Reduction Factor	2.59	7.85	13.32	22.84	117.64	391.94	1638.38
Trigger Efficiency	0.98	0.96	0.95	0.94	0.91	0.88	0.85
Trigger Purity	0.90	0.95	0.94	0.93	0.87	0.79	0.68
Rate at 800 Hz (Hz)	308.38	101.97	60.05	35.02	6.80	2.04	0.49

Table 6.3: Summary results for a high-p_t single particle trigger.

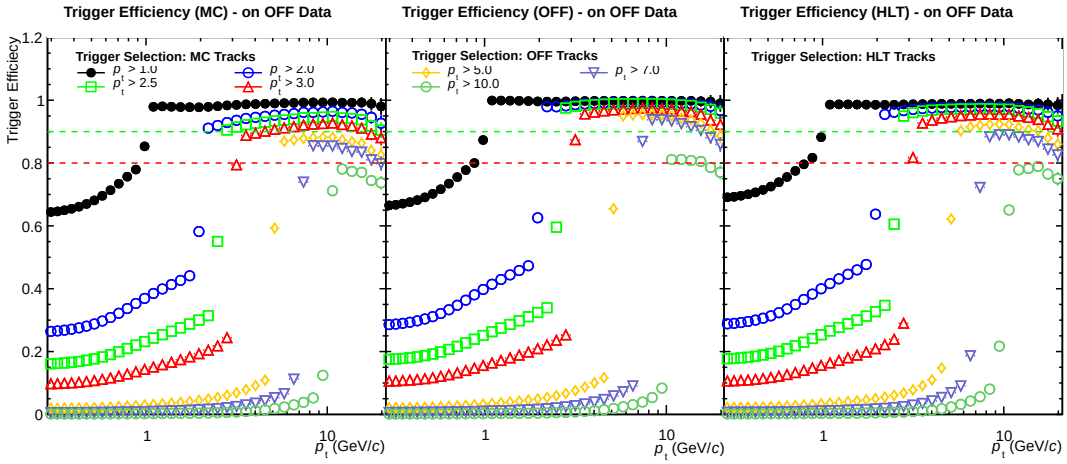

Figure 6.9: The efficiency of the p_t distribution of the off-line reconstructed tracks for different trigger inputs and different trigger thresholds. This figure shows from left to right the efficiency of the p_t distribution of the off-line reconstructed tracks with an applied trigger with of Monte Carlo particles, off-line reconstructed tracks and HLT reconstructed tracks. Different colors indicate different trigger thresholds.

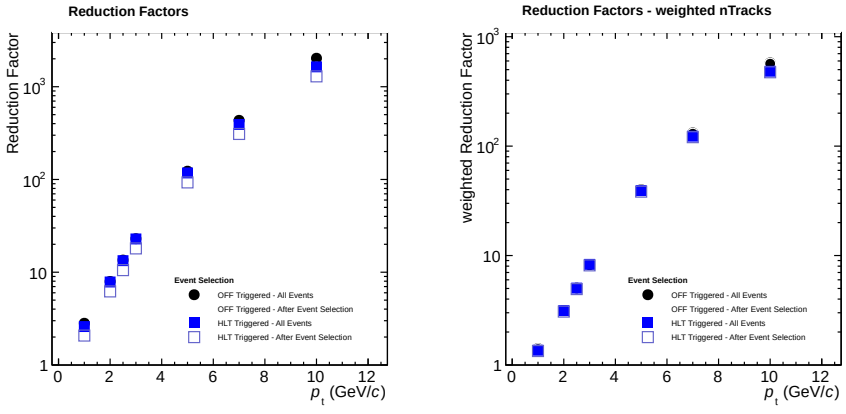

Figure 6.10: The reduction factor for different trigger thresholds. The left figure shows the reduction factor for different p_t thresholds for all events and those, which have passed the event selection. The figure on the right side shows the same quantities as on the left side, but the reduction factor is weighted with the number of track in each event .

7. Conclusions

An important milestone has been reached with the completion of the successful commissioning of the ALICE HLT in spring 2010 with the first pp collisions at $\sqrt{s} = 7\,\text{TeV}$. This thesis introduced and motivated the necessity of an HLT in the ALICE data-taking process and described its development and implementation. The most important aspects of the hardware and software commissioning as well as of the integration into ALICE have been discussed. Furthermore, the motivation for and implementation of a trigger on high momentum particles have been given.

ALICE was designed to cope with the extreme scenario of a charged particle multiplicity density of $\mathrm{d}N_{\text{ch}}/\mathrm{d}\eta = 8000$ at a read-out rate of 200 Hz central Pb–Pb collisions. For pp collisions the anticipated rate was 1 kHz [63]. The need for data reduction with an HLT, as it has been presented in section 3.2.1, manifests itself in the read-out data volume, as well as in the available storage bandwidth and space. In order to achieve this data reduction, the means of triggering and data compression are applied.

The HLT has been built as a high-performance compute cluster using abstraction layers to separate hardware, data-transport, and reconstruction and analysis software. In order to achieve the design performance, different means of data parallelization on the data-transport level as well as in the reconstruction software have been implemented. The HLT has been fully integrated into the ALICE data path and is connected to the other on-line systems via its interfaces.

The commissioning process and, therefore, also the realization of the HLT, has been described. Moreover, the general implementation of the HLT cluster, as well as detailed descriptions of important aspects of the HLT commissioning have been discussed. Achieving processing rates of up to 1.5 kHz and event sizes up to 1.5 MByte in pp collisions, the HLT has not only fulfilled the design requirements [63], but also exceeded them. The main achievements described in this thesis can be highlighted as follows:

- Building and commissioning of a working and performing HLT prototype
- Building up of an high performance compute cluster
- Design and commissioning of the configuration and RunControl of the HLT system
- Design and implementation of application software distribution mechanisms
- Integration of the ALICE detectors and the interfaces to the other on-line and off-line systems
- Provision of on-line reconstructed data

- Development of the first trigger application in pp collisions, a high-p_{t} particle trigger

It is a challenge to test and benchmark such a large-scale complex system, if no development system of a similar size exists. The development cluster could only play a limited role for basic performance and running tests. In order to test scalability of the HLT in its full size, the production cluster had to be used to perform tests and benchmarks, as the most issues during the commissioning only arose in the interplay with the other on-line systems.

Such tests, including all on-line systems and detectors, where extremely difficult to set up and accomplish. In order to have a complete test of the whole data-taking functionality, a fully running system needs to be available. During the commissioning phase of ALICE this was not always the case, as first of all the priority was given to the commissioning of the detectors themselves. Later on, these full data-taking tests were very often interfering with ALICE activities such as cosmic data-taking or other global activities. Only 10 months after the first collisions (after the work for this thesis) a new run species was introduced in ALICE, a TECHNICAL run without the pressure of physics data-taking. This helped to get the detectors under control as the data-taking functionality could be mimicked. Nevertheless, this is only a very basic test for the on-line systems as the events which are sent through the data-taking chain are almost empty and the trigger mix does not reflect the reality. The test procedure for the complete data-flow has not been the highest priority during the commissioning of ALICE and has to be further improved in future, in order to increase the data-taking efficiency.

As the HLT cluster is the first high performance compute cluster designed, developed, and built by this group of people, several lessons had to be learned during its commissioning. Some of these points then have been taken in to account by follow up projects carried out by some of the group. But even during the commissioning itself, issues concerning the hardware and software infrastructure, were investigated and the solutions found have been implemented in the SysMES system to allow for automatic reactions, in order to stabilize the full HLT. Also some design considerations have been proven as suboptimal, such as concentrated GigaBit-Ethernet switches in two racks and the use hardware from the blade of the cutting edge.

Further extensive benchmarks and systematic checks in order to bring the HLT in to a stage being able to process heavy-ion collisions have been beyond the scope of this thesis. However, they have been carried out, not by the author, since then and were companioned by further hardware extensions. The data-network has been completely switched to InfiniBand and the final number of compute nodes has been purchased and installed, including GPUs for the on-line TPC tracking components.

The trigger on high-p_{t} particles was developed to be the first real application during pp data-taking. It was pushed forward by the potential need of it during the second half of the 2010 data-taking period, in order to save storage space. Using already recorded pp as well as simulated pp collisions, the trigger was tested and verified. However, the LHC then went through a longer development phase, which allowed ALICE to run without triggering in the first year and made sure the storage limitations were met.

The first real application after the commissioning phase, which has been carried out by the HLT system, besides data quality monitoring, is the compression of the

TPC raw data using means of on-line reconstruction. Since the 2011 Pb–Pb run on-line reconstructed TPC clusters are stored to reduce the data volume by a factor of 4 and to be used later in the off-line reconstruction.

A. Trigger Naming

Several trigger classes and trigger menus will be created in the life-time of the HLT. In order to keep this information structured in a systematic way, a naming convention was defined on the basis of the naming schema of the CTP trigger classes (see also [138]).

A.1 HLT Trigger Menu

An *HLT Trigger Menu* is a list of *HLT Trigger Classes* with associated pre-scaling factors. A given HLT trigger menu is defined by an *HLT TriggerMenuIdentifier*.

`HM-<NAME>-V<XXXX>`

`<NAME>` A meaningful, human readable Name.

`<XXXX>` A version number, 4 digits.

Examples :

- `HM-COSMICS-V0001` : First version of a trigger menu for data-taking with cosmic particles.
- `HM-PP_HIGH_MULTIPLICITY-V0002` : Second version of a high multiplicity trigger for pp collisions.

An HLT trigger menu is described by two complementary configuration objects:

- A text-file within the RunControl configuration schema contains the possible **input trigger components**. Their trigger decisions are then combined in the HLT global trigger. Details about the RunControl configuration schema can be found in section 5.5.2.
- As every HLT component, also the **HLT global trigger** (implemented in the `AliHLTGlobalTriggerComponent` class) is configured via a ROOT based configuration macro, which is stored in the HCDB. It contains the logic to combine the different trigger inputs in the HLT global trigger.

Note: Not all input trigger components have to be actually used in the configuration of the global trigger.

A.2 HLT Trigger Class

Every HLT trigger class can be a logical function of several CTP trigger classes and a complex function of analysis results. As the trigger algorithm in addition will evolve in time, a versioning for the trigger names was introduced. The HLT can filter on DDL granularity, which is impossible to encode in a name which should be human readable.

Therefore, a flag was included, which encodes whether a region-of-interest should be filtered or whole detectors. The detectors to be read out also have to be included. A generic HLT trigger class name is defined by:

`H-<NAME>-V<XXXX>.<YYY>-<DOMAIN-SELECTION>-<SELECTION-TYPE>-<PRIORITY-TAG>`

`<NAME>` A description of the trigger algorithm in a human readable form.

`<XXXX>` A major version number, encoding the exact trigger conditions and the associated CTP trigger classes, 4 digits.

`<YYY>` A minor version number of the trigger algorithm, 3 digits.

`<DOMAIN-SELECTION>` A trigger domain merging expression, referring to set of detectors to be triggered and read out.

`<SELECTION-TYPE>` Flag for the trigger selection type:

`ALL` Select full detectors

`ROI` Region-of-interest, single DDL granularity

The following optional element can be added :

`<PRIORITY-TAG>` A priority tag, which allows to reserve a special bandwidth exclusively for this trigger class. The possible values are: `P0`, `P1`, `P2`, and *none*, where `P0` has the highest priority

Examples :

- `H-HIGH_PT-V0013.001-CENT-ALL-P1` : A high-p_{t} trigger selecting all DDLs for the `CENT` subset of detectors in the central barrel with priority 1.
- `H-MULTIPLICITY_COSMICS-V001.001-ITS-ALL` : A multiplicity trigger selecting all DDLs for the `ITS` subset of detectors in the central barrel.
- `H-J_PSI_FAST-V009.002-CENT_FAST-ROI-P0` : A fast J/ψ trigger selecting a region-of-interst for the `CENT_FAST` subset of detectors in the central barrel with highest priority.

The trigger components themselves are configured via HCDB objects, which can interpret the class names as well as the major and minor version numbers.

A.3 HLT Trigger Menu for Cosmic Data-Taking Campaign

During the 2009 data-taking campaign of cosmic particles the HLT trigger menu HM-COSMICS-V0001 was used. The configuration for the input trigger components is given in Listing A.1 and the HCDB object to configure the global trigger is laid out in Listing A.2.

```
BarrelMultiplicityTriggerITS
BarrelGeomMultiplicityTriggerPHOS
```

Listing A.1: The input trigger components for HM-COSMICS-V0001

```
// -- ITS
config.AddItem("BarrelMultiplicityTriggerITS",
    "domainHLTOUT|domainALLDDL",
    "H-TRACK_MULTIPLICITY-V0001.001-CENTRAL_ITS-ALL");

// -- CTP POSITIVE LIST
config.AddItem("C0OCP-ABCE-NOPF-CENT",
    "domainHLTOUT|domainALLDDL",
    "H-CTP_TRUE-V0001.001-CENTRAL-ALL");
config.AddItem("CTRDCO2-ABCE-NOPF-CENT",
    "domainHLTOUT|domainALLDDL",
    "H-CTP_TRUE-V0002.001-CENTRAL-ALL");
config.AddItem("C0OBE-ABCE-NOPF-CENT",
    "domainHLTOUT|domainALLDDL",
    "H-CTP_TRUE-V0003.001-CENTRAL-ALL");

// -- TOF C0OB3 scale down
config.AddItem("C0OB3-ABCE-NOPF-CENT",
    "domainTPCDDL|domainTOFDDL|domainCTPDDL|domainHLTDDL|domainHLTOUT",
    100, "H-CTP_SCALE_DOWN-V0001.001-CENTRAL_TOF-ALL");
```

Listing A.2: The HLT gobal trigger configuration for HM-COSMICS-V0001

Two trigger input components are allowed by the trigger input configuration file. However, only the first one (`BarrelMultiplicityTriggerITS`) is used in the HLT global trigger configuration, which is identified by the unique identifier `HM-COSMICS-V0001` and the second one (`BarrelGeomMultiplicityTriggerPHOS`) is unused. The HLT global trigger configuration itself is divided in three logical parts, which are combined with a logical or for the final trigger decision:

ITS In the first part, the HLT trigger on cosmic particles is defined. The trigger decision is calculated in the HLT trigger component `BarrelMultiplicityTriggerITS` and the raw data of all detectors and the reconstructed HLT are stored, as specified by the trigger domain `domainHLTOUT|domainALLDDL`.

The trigger algorithm requires minimum one reconstructed track, crossing the fiducial volume of the SDD detector, and is identified by the trigger name `H-TRACK_MULTIPLICITY-V0001.001-CENTRAL_ITS-ALL`.

CTP The second part describes the positive list of three CTP trigger classes, which should always be stored by the DAQ, using the same trigger domain.

TOF In the third part, a special trigger for performance studies of the TOF is defined. The CTP trigger `C0OB3-ABCE-NOPF-CENT` is scaled down by a factor of 100 and only the subset of the raw data (TPC, TOF, CTP and HLT) is stored.

B. Network Naming

This chapter is adapted from the HLT internal wiki page [139] and included here for completeness.

B.1 IP Address Assignment Rules

The HLT inter-node communication is based on the IP protocol stack. Therefore, all nodes have to be identified accordingly with IP addresses. The assignment is based on the class B network 10.162.x.x, designated to the HLT from the CERN IT department. Therefore, there are 16 bits available, which can be defined for HLT internal usage.

Summary

All IP addresses are based on the sub-network and the position of the node within the HLT counting rooms, following this definition :

```
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
|0|0|0|0|1|0|1|0|  |1|0|1|0|0|0|1|1|  |N|N|S|S|W|W|R|R|  |R|R|R|O|O|O|O|O|
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
10               . 162              . x                  . x
```

With:

N defining a bit for the sub-**N**etwork division

S defining a **S**pare bit

W defining a bit for the ro**W** number/position

R defining a bit for the **R**ack position in a row

O defining a bit for the n**O**de position in a rack

Sub-Networks (2 bits)

4 subnets are used, which can be physically separated, but don't have to be. For the distinction of the subnets the top 2 bits are used. The subnet mask is thus 255.255.192.0 .

1. Bits 00 : 10.162.0.0 : Data Network, GigaBit-Ethernet Port 1 (ETH1), sub-network : 10.162.0.1-10.162.63.254
2. Bits 01 : 10.162.64.0 : Data Network, GigaBit-Ethernet Port 2 (ETH2), sub-network : 10.162.64.1-10.162.127.254
3. Bits 10 : 10.162.128.0 : Maintenance Network, Fast-Ethernet Port 0 (ETH2), sub-network : 10.162.128.1-10.191.254
4. Bits 11 : 10.162.192.0 : Backbone Network, InfiniBand over IP, sub-network : 10.162.192.1 - 10.162.254.254

Spare Bits (2 bits)

2 bits are kept as spares for later usage.

Node Addresses (12 bits)

The remaining 12 bits (W,R,O) are used to built a geographical addressing scheme for the nodes:

Row (2 bits) 2 bits are used to define the row of racks within the HLT counting rooms.

1. Bits 00 : CR2 Row X
2. Bits 01 : CR2 Row Y
3. Bits 10 : CR2 Row Z
4. Bits 11 : CR3 Row X

Rack Position (5 bits) 5 bits are used to define the rack number (for example CR2-Y07 has rack number 7 and row number 1). The maximum number of racks in a row is 17 for CR2 row Y.

Node Position in Rack (5 bits) 5 bits are be used to define the position of a node in a given rack (counting from 0 for the lowest position upwards). Using 3 U cases at most 17 nodes can be placed into a rack. Therefore, the node position in a rack is the base height unit of the computer divided by 3 for most racks.

Special cases

Since no row has more than 17 racks, the rack positions 18-31 are always unused, and therefore, a few pseudo racks are defined in that range. These addresses can be used for devices that are not assigned to a rack (e.g. IP webcams, virtual machines, ...), as well as for dynamic DNS, for instance:

Row CR2-X rack 31 (NN=00 SS=00 WW=00 RRRRR=11111): dynamic DNS i.e. 10.162.3.224 ... 10.162.3.255

B.2 ETHn Port Assignment Rules

The index of the ETHn port numbering is defined logically according to the table below. As a consequence many machines will not have an ETH0, port but ETH1 and ETH2 .

ETH0: Reserved for special purpose network ports or node management interfaces (typically Fast-Ethernet)

ETH1: HLT network connection - all nodes (GigaBit-Ethernet)

ETH2: Trunked with ETH1 or external port of portal nodes (GigaBit-Ethernet)

B.3 IP Monitoring Proxy Port Assignement Rules

External Ports on Portal Nodes In order to allow remote access to the various monitoring TCP Dump Subscribers potentially running on the nodes of the HLT appropriate proxies are installed on the portal nodes vobox0/vobox1 and dcs0/dcs1. In order to simplify the number allocation the port numbers follow the IP address assignment of the nodes. The various monitoring sources can be easily merged on one node without overhead. It is sufficient to have a small number of TCP Dump Subscribers, possibly with different filter characteristics.

The following numbering scheme is implemented:

```
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
|1|1|W|W|R|R|R|R|  |R|O|O|O|O|O|I|I|
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
```

With:

W defining a bit for the ro**W** number/position

R defining a bit for the **R**ack position in a row

O defining a bit for the n**O**de position in a rack

I defining the TCP Dump Subscriber **I**D (0–3)

Correspondingly, the IP Port numbers occupy the upper quarter (16384 codes) of the available 16-bit number range, starting at 49152 and reaching until 65535.

Internal Ports on HLT Nodes Internally, no proxies are needed so that the destination port numbers on the HLT internal nodes are always the same (49152–49155):

```
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
|1|1|0|0|0|0|0|0|  |0|0|0|0|0|0|I|I|
+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+
```

With:

I defining the TCP Dump Subscriber **I**D (0–3)

C. Optical Fiber Connections

The 460 optical fibers (DDLs) from the DAQ to the HLT and 10 optical fibers from the HLT to the DAQ assure the exchange of physics data and HLT trigger decisions. The end-points of those optical fibers in between CR2 and CR1 (DAQ) are in optical patch-panels. Their position in CR2, the number of DDLs, as well as their detector assignment are described in the tables below.

Two types of optical fiber can be distinguished according to their position in the data flow.

- Fibers from DAQ-to-HLT, for the incoming connections
- Fibers from HLT-to-DAQ, for the outgoing connections

In case of failures of some fibers, 36 spare fibers, 12 in each row, have been put in place right from the beginning.

Rack	**# DDLs**	**Detector**
CR2-X2	12	TPC A00 A01
CR2-X3	12	TPC A02 A03
CR2-X4	12	TPC A04 A05
CR2-X5	12	TPC A06 A07
CR2-X6	12	TPC A08 A09
CR2-X6	12	Spare
CR2-X7	12	TPC A10 A11
CR2-X8	12	TPC A12 A13
CR2-X9	12	TPC A14 A15
CR2-X10	12	TPC A16 A17

Table C.1: Optical fiber positions - CR2 Row X - DAQ-to-HLT

Rack	# DDLs	Detector
CR2-Y5	36	TOF A
CR2-Y6	36	TOF B
CR2-Y7	18	TRD
CR2-Y9	12	TPC C16 C17
CR2-Y10	12	TPC C14 C15
CR2-Y10	12	Spare
CR2-Y11	12	TPC C12 C13
CR2-Y12	12	TPC C10 C11
CR2-Y13	12	TPC C08 C09
CR2-Y14	12	TPC C06 C07
CR2-Y15	12	TPC C03 C07
CR2-Y16	12	TPC C02 C05
CR2-Y17	12	TPC C00 C03

Table C.2: Optical fiber positions - CR2 Row Y - DAQ-to-HLT

Rack	# DDLs	Detector
CR2-Z2	24	EMCAL
CR2-Z3	36	SPD/SSD
CR2-Z5	24	SDD
CR2-Z6	8	Trigger Detectors
CR2-Z6	12	Spare
CR2-Z7	20	HMPID/PMD
CR2-Z10	20	PHOS
CR2-Z12	22	DiMUON

Table C.3: Optical fiber positions - CR2 Row Z - DAQ-to-HLT

Rack	# DDLs	Detector
CR2-Z8	10	HLTOUT

Table C.4: Optical fiber positions - CR2 Raw Z - HLT-to-DAQ

D. Cluster Hardware Description

This chapter gives an overview of the technical details of the installed PC and switch hardware in the HLT cluster.

D.1 Cluster Nodes

The HLT cluster nodes are grouped by their function in three different categories: Front-End Processors, Computing Nodes, and Infrastructure Nodes. However, they are a heterogeneous group of machines, consisting of 6 different types, identified by their motherboard type:

gw-type Tyan® Thunder K8S Pro (S2882)

vobox-type Supermicro H8DCi

ms-type Tyan® Thunder K8WE (S2895)

fep-type Tyan® Thunder H2000M (S3992)

cn1-type Tyan® Tempest i5400PW (S5397)

cn2-type Supermicro X8DTT

A detailed description of hardware setup of the nodes is given below.

D.1.1 Front-End Processors (FEPs)

The HLT production environment consists of 121 FEP nodes (fep-type), where 118 are connected to optical-fibers and 3 are used as hot-spares, as laid out in Table D.2. Additionally, another 12 FEP nodes are part of the development environment. The 3 U chassis are all equipped with the motherboard Tyan® Thunder h2000M (S3992), two Quad Core AMD Opteron™ 2378, 2.4 GHz processors, 12 GByte of RAM and two GigaBit-Ethernet network adapters, as listed in Table D.1.

They host two PCI-X based H-RORCs each, allowing for a maximum of 4 DDL connections per node. Having the two PCI-X slots on two separate PCI-X buses, was the key requirement when choosing the actual motherboard. This guarantees enough bandwidth for the physics data to be written from the H-RORC into the node's main memory.

Node management is done via extra added PCI cards, being either a CHARM[1] card [100] or a proprietary BMC[2] controller card.

[1] Computer-Health-And-Remote-Monitoring
[2] Baseboard Management Controller

The hostnames of the FEP nodes follow the general schema `fep<detector><ID>`. However, several exceptions to that schema exist. Especially some smaller detectors are grouped in one physical machine. The TPC as biggest detector is divided according to its physical layout in A and C side, as well as sector-wise in inner (2 DDLs per sector) and outer read-out partitions (4 DDLs per sector), where always the inner read-out partitions of tow adjacent sectors are combined in one node.

# Nodes	**Motherboard Type**	**Processor Type (2 CPUs inside)**	**Memory (GByte)**
121	Tyan® Thunder h2000M (S3992)	Quad Core AMD Opteron™ 2378	12

Table D.1: Installed front-end processor nodes.

Sub-Detector	**# FEPs**	**# DDLs**	**Hostnames**
SPD	5	20	fepspd[0-4]
SDD	6	24	fepsdd[0-5]
SSD	4	16	fepssd[0-4]
TPC	54	216	feptpc[a,c]i[00,02,...,16], feptpc[a,c]o[00,01,...,17]
TRD	5	18	feptrd[00,04,08,10,14]
TOF	18	72	feptof[a,c][0,8]
HMPID	3.5	14	fephmpid[0,3]
PHOS	5	20	fepphos[0,4]
PMD	1.5	6	feppmd, fephmpid3
MuonTRK	5	20	fepdimutrk[0,4]
MuonTRG	1	2	fepdimutrg
FMD	0.75	3	fepfmdacorde
T0	0.25	1	feptriggerdet
V0	0.25	1	feptriggerdet
ZDC	0.25	1	feptriggerdet
ACORDE	0.25	1	fepfmdacorde
CTP	0.25	1	feptriggerdet
EMCAL	5	24	fepemcal[0-4]
HLT	3	10	fephltout[0-2]
Hot-Spare	3		fepspare[0-2]
Total	121	470	

Table D.2: FEP nodes dedicated to the sub-detectors and their hostnames. One node each of TRD, MuonTRG and HLT contain only two DDL links. For EMCAL only 20 DDL links are connected.

D.1.2 Computing Nodes (CNs)

Two different versions of CN nodes (cn1-type and cn2-type) exist and are listed in Table D.3. The early version (cn1-type) is equipped with the Tyan® Tempest i5400PW(S5397) motherboard, two Intel® Xeon® E5420 (4 cores each), 2.5 GHz processors, 16 GByte of RAM and has a 3 U chassis. 15 of those nodes are used in the production and 6 in the development environment. The production nodes serve additionally as so-called IB-routing nodes (see section 5.3.2) and therefore, they host additionally to the two GigaBit-Ethernet network adapters a QDR IB network adapter. Furthermore, they are equipped with a hardware PCIe[3] SATA2[4] RAID controller card and eight 750 GByte hard disks. The node management is done via extra added CHARM cards.

A more condensed chassis architecture was chosen for the later generation (cn2-type), having four motherboards implemented in two height units (see Figure D.1). Nine chassis with a total of 36 nodes have been installed. The Supermicro X8DTT-IBX motherboards are arranged in two hot-swappable drawers on each height unit. They are equipped with two Intel® Xeon® E5520 (2x4 cores each), 2.27 GHz processors and 24 GByte of RAM. Furthermore, they have two GigaBit-Ethernet network adapters, as well as one QDR IB network adapter, and a BMC controller on-board.

# Nodes	Motherboard Type	Processor Type (2 CPUs inside)	Memory (GByte)
15	Tyan® Tempest i5400PW(S5397)	Intel® Xeon® E5420, 2.5 GHz	16
36	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24

Table D.3: Two generations of computing nodes have been installed.

D.1.3 Infrastructure Nodes

The infrastructure nodes consist of all 6 different node types. All nodes except the gw-type nodes are equipped with IB network adapters. Nodes of the cn2-type have them on-board, whereas the others are equipped with PCIe IB network adapter cards. The node management for all but cn2-type nodes is done with CHARM cards, hence cn2-types nodes are equipped with an on-board BMC controller.

Storage/Development Nodes The storage nodes are listed in Table D.4 and are all equipped with hardware PCIe SATA2 RAID 6 controllers. Additionally, two of the development nodes (dev2 and dev3) host storage for uncritical data. The main storage nodes ms0/ms1 are equipped with 8 x 1 TByte hard disks each, ms2/ms3 with 8 x 500 GByte hard disks each, and dev2/dev3 with 8 x 750 GByte hard disks each. This is a total of 36 TByte disk capacity from which 27 TByte are usable in a RAID 6 system. The DB nodes db1/db2 host 8 x 300 GByte fast access hard disks each.

Gateway/Portal Nodes The gateway and portal nodes listed in Table D.5 allow for access from outside into the HLT cluster and vice versa. Their Ethernet network interface eth2 is used to connect to the other ALICE and CERN networks, while eth1 is

[3] PCI Express
[4] Serial ATA, Serial Advanced Technology Attachment version 2.0

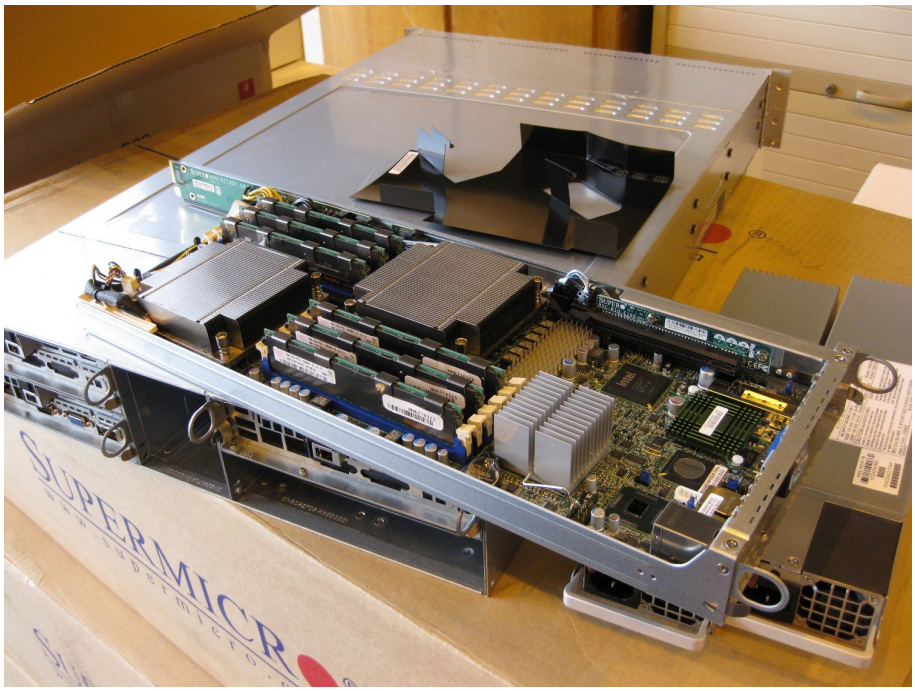

Figure D.1: A CN node of the second generation (cn2-type) demonstrating the modular design of the chassis.

their internal HLT connection. Users can only access the HLT cluster via the gw0/gw1 machines, however, they are directly forwarded to the development servers, as laid out in appendix E. The portal nodes (dcs0/dcs1, ecs0/ecs1, and vobox0/vobox1) are exclusively used for the interfaces.

Development Nodes The development servers, as listed in Table D.6, are the main development and compile nodes for the users, in addition to the development cluster, which is built out of fep-type, cn1-type, and cn2-type nodes.

Monitoring/GUI Nodes The mon0/mon1 nodes and their development nodes mondev0/mondev1 host the servers for SysMES and other monitoring applications, whereas the operator GUI application is hosted on the gui0/gui1/gui2 nodes, see also Table D.7.

Host	**Motherboard Type**	**Processor Type (2 CPUs inside)**	**Memory (GByte)**
ms0	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
ms1	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
ms2	Tyan® Thunder K8WE (S2895)	Dual Core AMD Opteron™ 265, 1.8 GHz	4
ms3	Tyan® Thunder K8WE (S2895)	Dual Core AMD Opteron™ 265, 1.8 GHz	4
db0	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
db1	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24

Table D.4: The mass storage and database nodes of the HLT cluster, consisting of ms-type and cn2-type nodes.

Host	Motherboard Type	Processor Type (2 CPUs inside)	Memory (GByte)
gw0	Tyan® Thunder K8S Pro (S2882)	AMD Opteron™ 242, 1.6 GHz	3
gw1	Tyan® Thunder K8S Pro (S2882)	AMD Opteron™ 242, 1.6 GHz	3
dcs0	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	3
dcs1	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	4
ecs0	Tyan® Thunder K8S Pro (S2882)	AMD Opteron™ 242, 1.8 GHz	2
ecs1	Tyan® Thunder K8S Pro (S2882)	AMD Opteron™ 242, 1.8 GHz	2
vobox0	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	4
vobox1	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	4

Table D.5: The gateway and portal nodes into the HLT cluster, consisting of gw-type and vobox-type nodes.

Host	Motherboard Type	Processor Type (2 CPUs inside)	Memory (GByte)
dev0	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
dev1	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
dev2	Tyan® Thunder H2000M (S3992)	Quad Core AMD Opteron™ 2346 HE, 1.8 GHz	8
dev3	Tyan® Thunder H2000M (S3992)	Quad Core AMD Opteron™ 2346 HE, 1.8 GHz	8

Table D.6: The head nodes for user login and development, consisting of cn2-type and cn1-type nodes.

D.2 Infrastructure Switches

The HLT cluster is made up of several networks as described in section 5.3.2 and, therefore, utilizes different kind of switches, as laid out in Table D.8.

The Fast-Ethernet maintenance network consists of 39 24-port Netgear FSM726 switches, the so-called charm switches, used for the node management. 26 48-port HP ProCurve switches build a treelike structure for the GigaBit-Ethernet data network. A 10 GigaBit-Ethernet optical transceiver is built in the 24-port HP ProCurve switch (SWGPN) as connection to the CERN GPN network. The Backbone QDR InfiniBand network has again a treelike structure and is built out of four 36-port Mellanox MTS3600Q switches.

D.3 Infrastructure External Connections

The HLT cluster is designed as an autonomous system, which can be reached over its gateway and portal nodes. Externally, they are identified with the 'alihlt-' prefix and '.cern.ch' suffix (e. g. `alihlt-gw0.cern.ch`). Table D.9 lists all external connections form the HLT cluster to different CERN networks.

In order to have redundant access from the CERN GPN to the HLT cluster, the gateway node `alihlt-gw0.cern.ch` is directly connected to an 100 MByte/s (Fast-Ethernet) wall outlet. All other connections to the GPN network via gw1 and vobox0/vobox1 are bundled in one switch (SWGPN), which has an 10 GigaBit-Ethernet (10 GByte/s) up-link.

Host	Motherboard Type	Processor Type (2 CPUs inside)	Memory (GByte)
mon0	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
mon1	Supermicro X8DTT	Intel® Xeon® E5520, 2.26 GHz	24
mondev0	Tyan® Tempest i5400PW (S5397)	Intel® Xeon® E5420, 2.50 GHz	16
mondev1	Tyan® Tempest i5400PW (S5397)	Intel® Xeon® E5420, 2.50 GHz	16
gui0	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	4
gui1	Supermicro H8DCi	Dual Core AMD Opteron™ 265, 1.8 GHz	4
gui2	Tyan® Thunder K8WE (S2895)	Dual Core AMD Opteron™ 265, 1.8 GHz	8

Table D.7: The monitoring and system management nodes for SysMES, the operator GUI, and other monitoring applications, consisting of cn2-type, cn1-type, vobox-type, and ms-type nodes.

Switch Type	# Ports	Network	# Switches
Netgear FSM726 v2	24	Fast-Ethernet	39
HP ProCurve 3400cl-24G (J4905A)	24	GigaBit-Ethernet	1
HP ProCurve 3400cl-48G (J4906A)	48	GigaBit-Ethernet	4
HP ProCurve 3500yl-48G (J8693A)	48	GigaBit-Ethernet	22
Mellanox MTS3600Q-1UNC	36	QDR InfiniBand	4

Table D.8: Installed Fast-Ethernet, GigaBit-Ethernet, and InfiniBand switches.

The dcs0/dcs1 portal nodes are connected via the patch-panel 3X6 to a switch in the DCS network, which at the same time is used to reach the ACR network for operational control.

A private node-to-node connection is used to connect the ecs0/ecs1 nodes directly to the two ECS head nodes in the DAQ/ECS network via the patch-panel 1Z4.

Internal Name	External Name	Connecting Network	External Connection
gw0	alihlt-gw0.cern.ch	GPN	Outlet 2222 P2-0000 0823/01
gw1	alihlt-gw1.cern.ch	GPN	Switch SWGPN
vobox0	alihlt-vobox0.cern.ch	GPN	Switch SWGPN
vobox1	alihlt-vobox1.cern.ch	GPN	Switch SWGPN
dcs0	alihlt-dcs0.cern.ch	DCS	Patch-panel 3x6
dcs1	alihlt-dcs1.cern.ch	DCS	Patch-panel 3x6
ecs0	alihlt-ecs0.cern.ch	DAQ/ECS	Patch-panel 1Z4
ecs1	alihlt-ecs1.cern.ch	DAQ/ECS	Patch-panel 1Z4
SWGPN		GPN	Outlet 2222 P2-Y401 Y401/01

Table D.9: The external connections from the HLT cluster to different CERN networks.

E. Cluster Access

Different ways to access the HLT cluster and the processed data exist, dependent on the originating network. A schematic overview is laid out in Figure E.1.

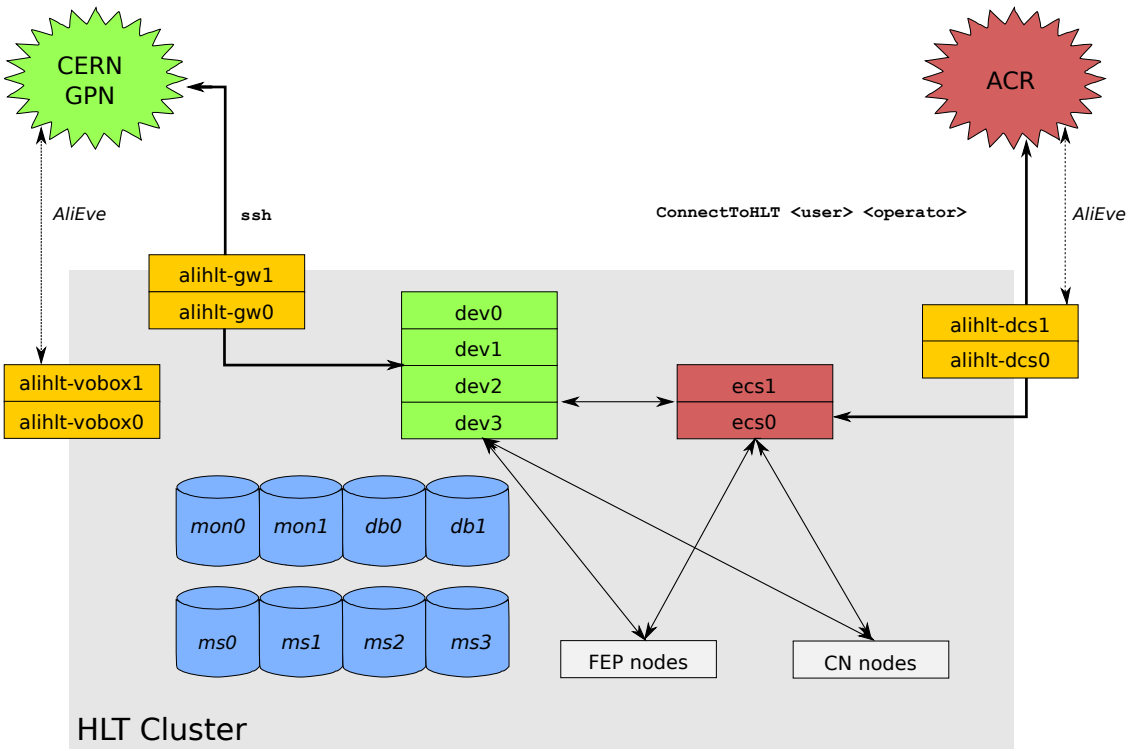

Figure E.1: Schematic overview of the user access to the HLT cluster.

E.1 User Access

Normal users in the CERN GPN network can access the HLT cluster via the gateway machines `alihlt-gw0.cern.ch` and `alihlt-gw1.cern.ch` using the ssh protocol[1]. A port forwarding allows the users to end up directly on the development nodes, e. g. `ssh -X -p 122 alihlt-gw0.cern.ch` leads to the dev0 node. The port mapping is given in Table E.1.

The *ALICE Control Room* (ACR) can be reached from the DCS network. Therefore, the operator users connect via the `alihlt-dcs0.cern.ch` and `alihlt-dcs1.cern.ch` nodes to the HLT cluster, which is described in detail in appendix E.2.

[1] Secure Shell (SSH)

Gateway Port	Destination Node
122	dev0
222	dev1
322	dev2
422	dev3

Table E.1: Access ports to the development nodes.

E.2 Operator Access

An *operator* user is a UNIX user in whose context all necessary processes for the HLT running are executed. Each operator implements its own running environment, a so-called *partition*.

As described in section 5.5.2, these operator user accounts have been created to allow for several parallel, instances of the HLT. In order to achieve this parallelization, every operator needs a unique port range, which can be used exclusively for the communication of the RunManager, as well as between different processes and TaskManagers. Furthermore every operator has its own standard settings and HCDB location. The existing operator accounts are listed in Table E.2.

Note : Within this thesis *operator* and *partition* are used synonymously.

Operator	Partition	RunManager Base Port	TaskManager Port Range	HCDB Location
hlt-operator	ALICE	20000	20100 - 21999	/opt/HCDB
tpc-operator	TPC	22000	22100 - 23999	$HOME/HCDB
trd-operator	TRD	24000	24100 - 25999	$HOME/HCDB
phos-operator	PHOS	26000	26100 - 27999	$HOME/HCDB
dimuon-operator	DIMUON	28000	28100 - 29999	$HOME/HCDB
its-operator	ITS	30000	30100 - 31999	$HOME/HCDB
dev-operator	DEV	32000	32100 - 33999	$HOME/HCDB
test-operator	TEST	34000	34100 - 35999	$HOME/HCDB
emcal-operator	EMCAL	36000	36100 - 37999	$HOME/HCDB

Table E.2: A table of the existing HLT operator users, their associated partition, port ranges and HCDB location.

HLT cluster nodes can be shared between operators. However, it is not wise to share FEP nodes, as their connected DDLs can only be used exclusively by an operator at a time.

The DAQ environment implements, similar to the HLT, also different partitions, which are used in the same, independent way as the HLT partitions. Any HLT partition can be connected to the any DAQ partitions as described in section 5.5.2.

In order to switch from the own UNIX user to the operator user account the command `Be<operator-name>` (e. g. BeTPC-operator) has to be executed. The privileges to do so are checked using the access control lists of the afs file-system.

From the operator station in the ACR network, the command

```
ConnectToHLT <user-name> <operator-name>
```

allows any privileged HLT user to connect as any operator user to the HLT cluster. Internally, the `Be<operator-name>` command is called and the operator is automatically redirected from the dcs0/dcs1 portal nodes to the head steering nodes ecs0/ecs1, where the running environment is automatically setup.

E.3 On-line Data Access/Event Display

In order to access HLT reconstructed data on-line or to use the HLT event display (AliEVE), no HLT user account is needed. Any ALICE member can access this information from any CERN network. The `AliHLTHOMERManager` application automatically takes care to connect to the appropriate portal node, as described in Table E.3.

Network	**Portal Node**
GPN	alihlt-vobox0.cern.ch / alihlt-vobox1.cern.ch
ACR/DCS	alihlt-dcs0.cern.ch / alihlt-dcs1.cern.ch

Table E.3: Portal nodes for on-line data access and the event display.

F. HLT Running States

This chapter is adapted and extended from the HLT internal wiki page [139] and included here for completeness. It describes the states of the HLT–ECS proxy and the TaskManager, as well as the mapping between the two.

HLT–ECS proxy states and transition commands use capital letters, where stable states are indicated by "STABLE_STATE", transitional states by *"TRANSITIONAL_STATE"*, and commands by *COMMAND*. For the TaskManager states the same notation applies, however, all in small letters.

F.1 ECS States

This section describes the HLT states of the ECS state machine, as they are transmitted from the ECS to the HLT. An overview of all the states is given in Figure F.1. Two different types of states are distinguished: stable states in the middle column and transitional, intermediate states in the left and right columns. The transitional states are triggered via explicit commands, except for the "ERROR" state, which is reached automatically in case of an error. The right column describes transitional states towards the "RUNNING" state, while the states on the left side are the transitional states on the reverse way towards the "OFF" state.

F.1.1 Detailed State Description

"OFF" In this state everything of the HLT is off. The HLT–ECS proxy and the RunManager wait for commands. With the *INITIALIZE* command, the state is changed to *"INITIALIZING"*.

"INITIALIZING" Currently nothing is done in this state, it is foreseen for future tasks, e. g. preparation of the on-line environment after a longer period without data-taking. The transition to "INITIALIZED" is performed automatically.

"DEINITIALIZING" As in the *"INITIALIZING"* state, nothing is currently done, but allows for future clean-up tasks for a clean transition into the "OFF" state. This state can be accessed either from the state "INITIALIZED" or from the "ERROR" state (from both via the *SHUTDOWN* command).

"INITIALIZED" Currently this state is identical to the "OFF" state.

"CONFIGURING" The *CONFIGURE* command provides the configure parameters and sends the HLT from "INITIALIZED' to *"CONFIGURING"*. During this state, the configuration files are created and the PENDOLINOs are started, the HCDB is prepared

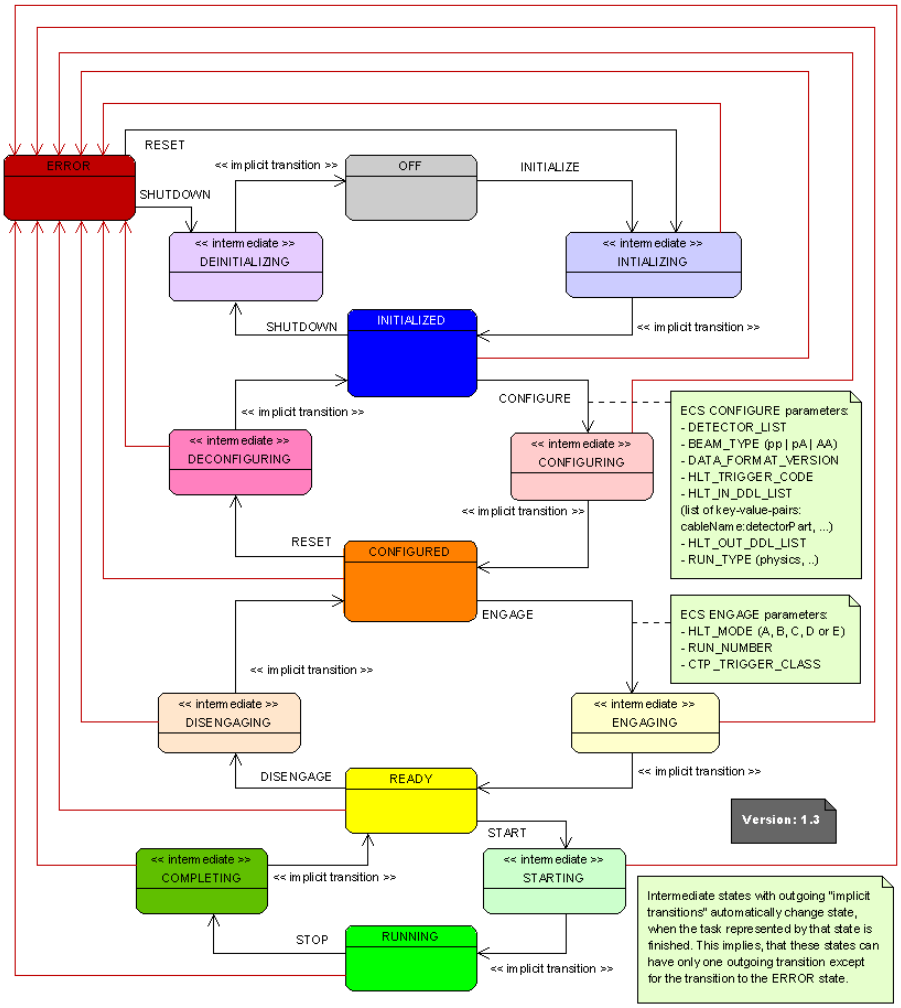

Figure F.1: State machine diagram of the HLT–ECS proxy. The stable states are located in the middle column, the right column shows the intermediate states from OFF to RUNNING, the left column the reverse direction. From [121].

and the TaskManager hierarchy is configured and the components are started. At the end of this state, all needed processes on all the nodes are started, and the transition to the state "CONFIGURED" is done automatically.

"DECONFIGURING" The transition from "CONFIGURED" to "INITIALIZED" is called *"DECONFIGURING"*. This transition is used to bring the HLT back to the ground state, to stop all components and the TaskMangers, and to disable the PENDOLINOs.

"CONFIGURED" The TaskManagers and components on each node are now started. With the *ENGAGE* command these components connect to each other, like it is foreseen in the configuration. To go back to the "INITIALIZED" state, the HLT–ECS proxy accepts a *RESET* command. To reconfigure the HLT, the HLT–ECS proxy needs also to return to the "INITIALIZED" state first, then a new *CONFIGURE* command can be issued. This is done automatically by the ECS.

"ENGAGING" This state is reached via the *ENGAGE* command, which also provides the current running conditions via the engage parameters. The started components first try to connect to each other locally on the same node and afterwards among different nodes. This may take some time and therefore this intermediate state is signaled to the ECS. When all connections are established "READY" is reached automatically.

"READY" The HLT system is now ready to start processing events. To begin with this task, the command *START* is needed. In order to leave current running conditions the *DISENGAGE* command can be used.

"STARTING" The transition state from "READY" to "RUNNING" is called *"STARTING"*. When all components are in the running state, and automatic transition to "RUNNING" is performed.

"RUNNING" Now, the HLT is running and is processing data. With a *STOP* command the HLT returns to the "READY" state.

"COMPLETING" After a run *"STOP"* signal from the ECS, it is most likely that some processes inside the HLT cluster may further run for quite some time to finish their data processing, which is indicated by this state. Only after the HLT–ECS proxy has returned to the "READY" state (implicit transition), the ECS can signal the offline Shuttle process to start requesting the FXS for calibration data.

"DISENGAGING" In order to leave the current running conditions from the READY state, the components of the have to disconnect from each other (locally and remotely). The system returns to the "CONFIGURED" state automatically and is ready for a new run.

"ERROR" The "ERROR" state can be reached from any state (except the OFF state). This state means, that something went wrong, and an internal error recovery of the HLT system failed. Then the system can be set in the "OFF" state with a *SHUTDOWN* command passing the intermediate state *"DEINITIALIZING"* or rebooted with a *RESET* command. In the later case, the state will transit to "INITIALIZED" passing the intermediate state *"INITIALIZING"*.

F.2 TaskManager States

The HLT–ECS proxy maps the ECS states to the states of the RunManager/TaskManager. Despite of the different names the TaskManager uses to indicate its states, some states have to be emulated by the HLT–ECS proxy as well as some states of the MasterTaskManager refer to just one state in the ECS. All mapped states and their corresponding transition commands are shown in Figure F.2. Correspondingly to the HLT–ECS proxy states, stable states are in the middle column, transitional states towards the "running" state are on the right side, and towards the "off/slaves dead" on the left side.

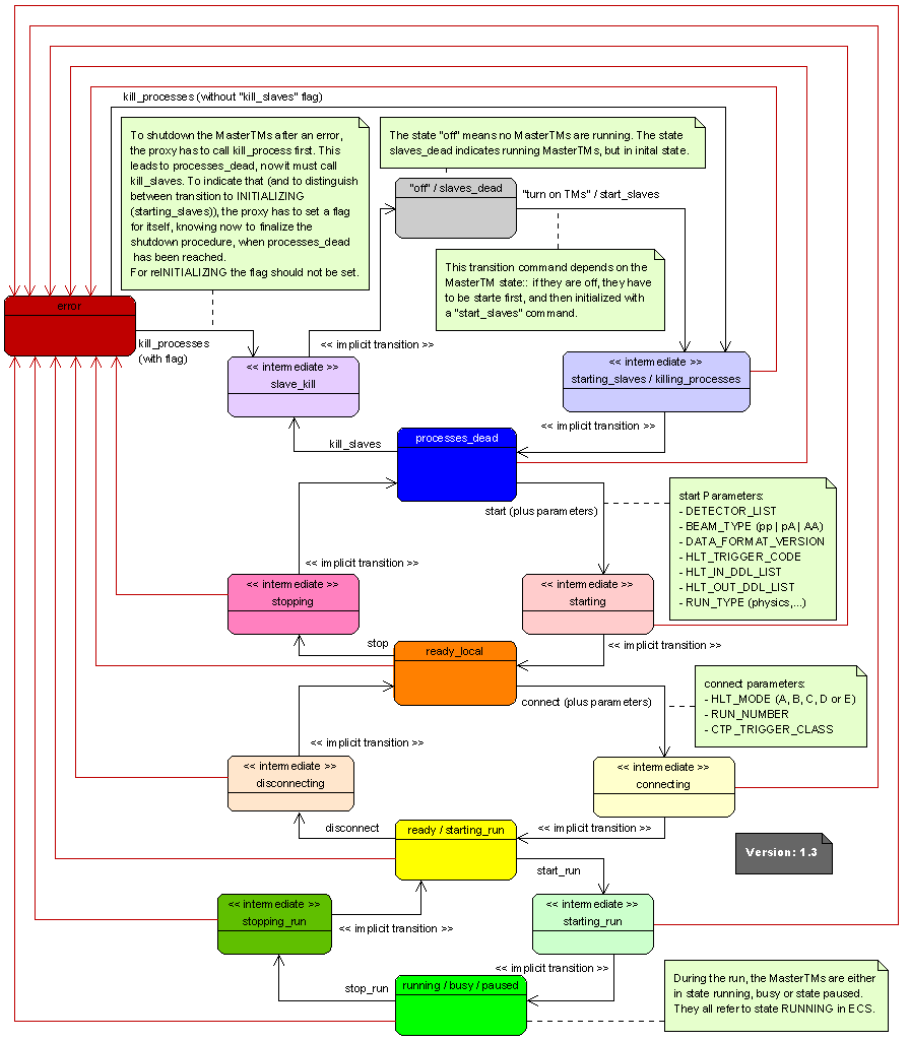

Figure F.2: State machine diagram showing the mapped HLT–ECS proxy states to the state names used by the TaskManager. Same colors as in Figure F.1 indicate the same state. From [121].

F.2.1 Detailed State Description

"off/slaves_dead" Refers to the "OFF" and "INITIALIZED" states in the HLT–ECS proxy and combines two TaskManager states. If the MasterTaskManager is really

"off", the proxy has to switch it on first. The state "slaves_dead" indicates a running MasterTaskManager, but in its initial state. In order to leave the "slaves_dead" state and the HLT–ECS proxy can call *start_slaves* as a part of the *CONFIGURE* command, which starts all Servant- and SlaveTaskManagers and ends up in the "processes_dead" state.

"processes_dead" With *kill_slaves* the MasterTaskManager will return to the "off/slaves_dead" state, stopping all Servant- and SlaveTaskManagers. To proceed with the *CONFIGURE* command, the *start* command has to be called, which now starts the processing components.

"ready_local" The "ready_local" state reflects the state "CONFIGURED" of the HLT–ECS proxy. To *ENGAGE* the HLT components, the HLT–ECS proxy issues the *connect* command, then the "READY" state will be reached. Going back to "processes_dead", the *stop* command can be issued (refers to the *RESET* from "CONFIGURED").

"ready" "READY" and "ready" are reflecting the same state. To transit to "RUNNING" the command *start_run* corresponds to *START*, *"DISENGAGING"* is performed via the *disconnect* command.

"running/busy/paused" The "RUNNING" state corresponds to three states of the TaskManager: the "running", the "busy", and the "paused" state. In the "running" state the HLT components waits for incoming data, while "busy" indicates that data is processed. The TaskManager can be also "paused", which allows for HLT internal error recovery. To *STOP* and return to the "ready" state (via *"COMPLETING"*), the HLT–ECS proxy triggers the *stop_run* command.

"error" "ERROR" and "error" are the same states.

G. RunControl Settings

This chapter describes the basic settings for the RunControl during the commissioning in the cosmics data-taking campaign.

.

RunManager Settings

```
RunManagerBasePort=${ALIHLT_PORT_BASE_RUNMGR}
MasterTaskManagerBasePort=${ALIHLT_PORT_BASE_TASKMGR}
Verbosity=0x79
TaskManagerDir=${ALIHLT_DC_DIR}
Production=1
ENABLEARCHIVE=0
ARCHIVETHISRUN=0
InfoLogger=/opt/HLT/tools/lib/libInfoLoggerLogServer.so
PendolinoHost=portal-dcs1
PendolinoManagerBasePort=${ALIHLT_PORT_PEND_MGR}
PendolinoCommandPath=${HLTPENDOLINO_TOPDIR}/RunManager-PendolinoWrapper.sh
PendolinoCount=1
PendolinoLogFile=/tmp/pendolino-starter.log
NoPendolino=0
```

Listing G.1: The RunManager settings, which have been used during the HLT commissioning

Globalsiteconfig Settings

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2Site>

<!-- ================================
                    Verbosity
     ================================ -->
   <!-- FORCED Verbosity -->
     <Verbosity>0x78</Verbosity>
   <!-- Forced Important, Fatal, Error, Warning -->
   <!-- EXCLUDED Verbosity -->
   <VerbosityExclude>0x07</VerbosityExclude>
   <!-- Forbids Debug, Benchmark and Information -->

<!-- ================================
                    DataFlow
     ================================ -->
   <EventMergerTimeout>10000</EventMergerTimeout>
```

```
    <!-- ms -->
    <!-- Activate dynamic merger timeout calculation, set maximum timeout -->
    <EventMergerDynamicTimeouts>50000</EventMergerDynamicTimeouts>
    <!-- ms -->
    <!-- Set minimum dynamic timeout value -->
    <EventMergerMinDynamicTimeout>10000</EventMergerMinDynamicTimeout>
    <!-- ms -->
    <!-- Activate dynamic input dis-/enabling and set maximum number of
         allowed successive events with missing contributions before an
         input is disabled. -->
    <EventMergerDynamicInputXabling>200</EventMergerDynamicInputXabling>
    <RunStopTimeout>10000</RunStopTimeout> <!-- ms -->
    <StopTimeout>10000</StopTimeout> <!-- ms -->

    <!-- <MaxEventAge>120</MaxEventAge> -->

  <!-- ===================================
                    Logging
       =================================== -->
    <SysMESLog>0</SysMESLog>
    <InfoLogger>/opt/HLT/tools/lib/libInfoLoggerLogServer.so</InfoLogger>
    <!-- <EventAccounting>/opt/HLT-public/test</EventAccounting> -->

  <!-- ===================================
            PreStart / PostTerminate
       =================================== -->
    <PreStartExec>ulimit -c unlimited</PreStartExec>
    <PostTerminateExec>
       ${HLTCONTROL_TOPDIR}/bin/system/post-exec-cmd.sh
    </PostTerminateExec>

  <!-- ===================================
                General Settings
       =================================== -->
    <TaskManagerDir>${ALIHLT_DC_DIR}</TaskManagerDir>
    <FrameworkDir>${ALIHLT_DC_DIR}</FrameworkDir>
    <ProductionRelease>1</ProductionRelease>
    <SubscriberRetryCount>10</SubscriberRetryCount>

  <!-- ===================================
             Shared Memory Settings
       =================================== -->
    <ShmLimit>bigphys:1000000000</ShmLimit>
    <ShmLimit>sysv:1000000000</ShmLimit>

    <NodeShmLimit>bigphys:4000000000</NodeShmLimit>
    <NodeShmLimit>sysv:7800000000</NodeShmLimit>

    <BridgeShmType>fephltout0:bigphys</BridgeShmType>
    <BridgeShmType>fephltout1:bigphys</BridgeShmType>
    <BridgeShmType>fephltout2:bigphys</BridgeShmType>

    <NodeShmLimit>fephltout0:bigphys:500M</NodeShmLimit>
```

```
   <NodeShmLimit>fephltout1:bigphys:500M</NodeShmLimit>
   <NodeShmLimit>fephltout2:bigphys:500M</NodeShmLimit>
</SimpleChainConfig2Site>
```

Listing G.2: The gobalsiteconfig settings, which have been used during the HLT commissioning, describing the basic settings for all HLT instances (Formatting adapted to fit in this work).

Siteconfig Settings

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2Site>
   <!-- FORCED Verbosity -->
   <Verbosity>0x78</Verbosity>
   <!-- EXCLUDED Verbosity -->
   <VerbosityExclude>0x07</VerbosityExclude>
   <!-- Set Event Scattering Parameters -->
   <EventModuloPreDivisor>4789</EventModuloPreDivisor>
   <ScattererParam>4789</ScattererParam>
</SimpleChainConfig2Site>
```

Listing G.3: The siteconfig settings, which have been used during the HLT commissioning, describing the basic settings for the HLT instance (Formatting adapted to fit in this work).

Globalddllist Settings

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2DDLList>

<!-- ##############################################
                     SPD
############################################### -->

   <DDL ID="0">
      <Node>fepspd0</Node>
      <Bus>3</Bus><Slot>4</Slot><Function>0</Function>
      <DIU>0</DIU>
   </DDL>
   <DDL ID="1">
      <Node>fepspd0</Node>
      <Bus>3</Bus><Slot>4</Slot><Function>0</Function>
      <DIU>1</DIU>
   </DDL>
   <DDL ID="2">
      <Node>fepspd0</Node>
      <Bus>2</Bus><Slot>5</Slot><Function>0</Function>
      <DIU>0</DIU>
   </DDL>
   <DDL ID="3">
      <Node>fepspd0</Node>
```

```
      <Bus>2</Bus><Slot>5</Slot><Function>0</Function>
      <DIU>1</DIU>
   </DDL>

   ...

</SimpleChainConfig2DDLList>
```

Listing G.4: A part of the globalddllist settings, which have been used during the HLT commissioning, describing the association of the ALICE unique DDL ID to a node, its H-RORC, and DIU. Only the first 4 of 470 DDL links are shown (Formatting adapted to fit in this work).

Nodelist Settings

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SimpleChainConfig2NodeList>

   <!-- ##############################################
                  MASTER NODES
        ############################################## -->
   <Node ID="portal-ecs0">
      <Hostname>portal-ecs0</Hostname>
      <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
      <Master>1</Master>
   </Node>
   <Node ID="portal-ecs1">
      <Hostname>portal-ecs1</Hostname>
      <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
      <Master>1</Master>
   </Node>

   <!-- ##############################################
                  FXS NODES
        ############################################## -->
   <Node ID="portal-vobox0">
      <Hostname>portal-vobox0</Hostname>
      <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
      <Attribute name="FXS" value="1"/>
   </Node>
   <Node ID="portal-vobox1">
      <Hostname>portal-vobox1</Hostname>
      <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
      <Attribute name="FXS" value="1"/>
   </Node>

   <!-- ##############################################
                  DCS NODES
        ############################################## -->
   <Node ID="portal-dcs0">
      <Hostname>portal-dcs0</Hostname>
      <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
```

```
    </Node>
    <Node ID="portal-dcs1">
       <Hostname>portal-dcs1</Hostname>
       <CPUs>4</CPUs><Platform>Linux-x86_64</Platform>
    </Node>

    <!-- ##########################################
                        SPD
         ########################################## -->
    <Node ID="fepspd0">
       <Hostname>fepspd0</Hostname>
       <CPUs>8</CPUs><Platform>Linux-x86_64</Platform>
    </Node>
    <Node ID="fepspd1">
       <Hostname>fepspd1</Hostname>
       <CPUs>8</CPUs><Platform>Linux-x86_64</Platform>
    </Node>
    <Node ID="fepspd2">
       <Hostname>fepspd2</Hostname>
       <CPUs>8</CPUs><Platform>Linux-x86_64</Platform>
    </Node>

    ...
</SimpleChainConfig2NodeList>
```

Listing G.5: A part of the nodelist settings of the *High-Level Trigger* (HLT) instance, which have been used during the HLT commissioning. It describes the setup of the setup up of the general steering and interface nodes, as well as the first FEP nodes of the SPD detector (Formatting adapted to fit in this work).

H. HOMER Manager Implementation

The definition of the four main classes of the HOMER Manager are laid out in this chapter. They are taken from revision 50000 of AliRoot trunk and can be found under `$ALICE_ROOT/HLT/BASE`. In order to fit in this work, the files have been slightly reformatted, but the C++ code has not been altered.

H.1 HOMER Source Descriptor

```
#ifndef ALIHLTHOMERSOURCEDESC_H
#define ALIHLTHOMERSOURCEDESC_H

/* This file is property of and copyright by the ALICE HLT Project
 * ALICE Experiment at CERN, All rights reserved.
 * See cxx source for full Copyright notice */

/** @file   AliHLTHOMERSourceDesc.h
    @author Jochen Thaeder
    @brief  Container for HOMER Sources */

/**
 * @defgroup alihlt_homer HOMER handling for AliROOT
 * This section describes the handling of HOMER Sources, Blocks
 * and the HOMER Reader inside the HLT and AliROOT  */

#include "TString.h"
#include "TNamed.h"

/**
 * @class AliHLTHOMERSourceDesc
 * This class contains the information of 1 homer source:
 * hostname, port for the HOMER interface as well as data
 * specifications. It used in order to fill these sources in TLists.
 * ( It has to inherit from TObject ). Furthermore, it knows if this
 * source was selected for read from a user.
 *
 * @ingroup alihlt_homer */
class AliHLTHOMERSourceDesc : public TNamed {

public:
  /**  constructor */
  AliHLTHOMERSourceDesc();
  /** destructor */
  virtual ~AliHLTHOMERSourceDesc();

  /** Set selection state
```

```
    * @param state      state, either kTRUE or kFALSE   */
  void SetState( Bool_t state ) { fSelected = state; }
  /** Checks if Source is selected to readout
    * @return           returns state, either kTRUE or kFALSE   */
  Bool_t IsSelected()           { return fSelected; }
  /** Select this source */
  void Select()                 { fSelected = kTRUE; }
    /** Deselect this source */
  void Deselect()               { fSelected = kFALSE; }

  /** Set Service of this source
    *  @param hostname  hostname of the source
    *  @param port      port of the source
    *  @param origin    HLT data origin
    *  @param type      HLT data type
    *  @param spec      HLT data specification */
  void SetService( TString hostname, Int_t port, TString origin,
                   TString type, TString spec );

  /** Get node name of this source
    * @return           hostname   */
  TString GetHostname()       { return fHostname; }
  /** Get node name of this source
    * @return           port   */
  Int_t   GetPort()           { return fPort; }
  /** Get name of this source
    * @return           name   */
  TString GetSourceName()     { return fSourceName; }
  /** Get detector of this source
    * @return           detector   */
  TString GetDetector()       { return fDetector; }
  /** Get sub detector of this source
    * @return           subdetector   */
  Int_t   GetSubDetector()    { return fSubDetector; }
  /** Get sub sub detector of this source
    * @return           subsubdetector   */
  Int_t   GetSubSubDetector() { return fSubSubDetector; }
  /** Get HLT data type of this source
    * @return           HLT data type   */
  TString GetDataType()       { return fDataType; }
  /** Get HLT specification of this source
    * @return           HLT specification   */
  ULong_t GetSpecification()  { return fSpecification; }

private:

  /** copy constructor prohibited */
  AliHLTHOMERSourceDesc(const AliHLTHOMERSourceDesc&);
  /** assignment operator prohibited */
  AliHLTHOMERSourceDesc& operator=(const AliHLTHOMERSourceDesc&);

  /** is selected to read out */
  Bool_t fSelected;
```

```
  /** Name of Source */
  TString fSourceName;

  // -- Service Specifications --
  /** Name of HOMER Node */
  TString fHostname;
  /** Name of HOMER port */
  Int_t fPort;

  // -- Data Specifications --
  /** HLT DataType */
  TString fDataType;
  /** Detector Name, e.g. PHOS, corresponds to HLT origin */
  TString fDetector;
  /** HLT Specification */
  ULong_t fSpecification;
  /** SubDetector Name e.g. MODULE */
  Int_t   fSubDetector;
  /** SubSubDetector Name e.g. PARTITION */
  Int_t   fSubSubDetector;

  ClassDef( AliHLTHOMERSourceDesc, 0 )
};
#endif
```

Listing H.1: The definition of the class `AliHLTHOMERSourceDesc` to be found in `HLT/BASE/AliHLTHOMERSourceDesc.h` (Formatting adapted to fit in this work).

H.2 HOMER Block Descriptor

```
#ifndef ALIHLTHOMERBLOCKDESC_H
#define ALIHLTHOMERBLOCKDESC_H

/* This file is property of and copyright by the ALICE HLT Project
 * ALICE Experiment at CERN, All rights reserved.
 * See cxx source for full Copyright notice  */

/** @file   AliHLTHOMERBlockDesc.h
    @author Jochen Thaeder
    @brief  Container for HOMER Blocks */

class AliHLTMessage;

#include "TString.h"
#include "TObject.h"
#include "AliHLTLoggingVariadicFree.h"

/**
 * @class AliHLTHOMERBlockDesc
 * This class contains the data which comes from 1 block, delivered
 * via the HOMER interface. It used in order to fill these block in
 * TLists. ( It has to inherit from TObject ). Furthermore, it reads
```

```
 * the specification of the block and classifies the data as TObject,
 * raw data, or something else.
 *
 * @ingroup alihlt_homer */
class AliHLTHOMERBlockDesc : public TObject, public AliHLTLogging {

public:
  /** standard constructor */
  AliHLTHOMERBlockDesc();
  /** destructor */
  virtual ~AliHLTHOMERBlockDesc();

  /** Set block data
   * @param data           Pointer to data
   * @param size           Size of data
   * @param origin         Detector
   * @param dataType       HLT data type
   * @param specification  HLT specification   */
  void SetBlock( void * data, ULong_t size, TString origin,
                 TString dataType, ULong_t specification );
  /** Returns if block contains a TObject
   * @return           Returns kTRUE if block contains a TObject,
   *                   kFALSE otherwise.   */
  Bool_t IsTObject()             { return fIsTObject;}
  /** Returns Pointer to TObject */
  TObject* GetTObject()          { return fTObject;   }
  /** Get class name of this block
   * @return           class name   */
  TString GetClassName()         { return fClassName; }
  /** Returns if block contains a raw data, fClassName is
   *  not set in this case.
   * @return           Returns kTRUE if block contains raw data,
   *                   kFALSE otherwise.   */
  Bool_t IsRawData()             { return fIsRawData;}
  /** Get pointer to data
   * @return           Pointer to data   */
  void* GetData()                { return fData; }
  /** Get size of data
   * @return           Size of data   */
  ULong_t GetSize()              { return fSize; }
  /** Get name of this block
   * @return           name   */
  TString GetBlockName()         { return fBlockName; }
  /** Get detector of this block
   * @return           Detector name   */
  TString GetDetector()          { return fDetector; }
  /** Get HLT data type of this block
   * @return           HLT data type   */
  TString GetDataType()          { return fDataType; }
  /** Get HLT specification of this block
   * @return           HLT specification   */
  ULong_t GetSpecification()     { return fSpecification; }
  /** Get sub detector of this block
```

```
   * @return           subdetector   */
  Int_t   GetSubDetector()        { return fSubDetector; }
  /** Get sub sub detector of this block
   * @return           subsubdetector   */
  Int_t   GetSubSubDetector()     { return fSubSubDetector; }

  /** Returns kTRUE if HLT specification indicates a
   *  subdetector range
   * @return           kTRUE if subdetector range   */
  Bool_t  HasSubDetectorRange()   { return fHasSubDetectorRange; }
  /** Returns kTRUE if HLT specification indicates a
   *  subsubdetector range
   * @return           kTRUE if subsubdetector range   */
  Bool_t  HasSubSubDetectorRange(){ return fHasSubSubDetectorRange; }

private:

  /** copy constructor prohibited */
  AliHLTHOMERBlockDesc(const AliHLTHOMERBlockDesc&);
  /** assignment operator prohibited */
  AliHLTHOMERBlockDesc& operator=(const AliHLTHOMERBlockDesc&);

  /** Set all additional members */
  void SetBlockParameters();
  /** Checks if Block contains a TObject.
   * If so, set fIsTObject to kTRUE, otherwise kFALSE
   * @return           fIsTObject   */
  Bool_t CheckIfTObject();
  /** Checks if Block contains a TObject raw data.
   * If so, set fIsRawData to kTRUE, otherwise kFALSE
   * @return           fIsRawData   */
  Bool_t CheckIfRawData();

  // -- Block properties --
  /** Pointer to data of the block */
  Char_t* fData;
  /** Size of data */
  ULong_t fSize;
  /** Block Name */
  TString fBlockName;

  // -- Data flags --
  /** States if block contains a TObject */
  Bool_t fIsTObject;
  /** States if block contains a raw data */
  Bool_t fIsRawData;

  // -- TObject properties --
  /** AliHTMessage object containg a TObject */
  AliHLTMessage* fMessage;
  /** TObject extracted out of @see AliHLTMessage */
  TObject*       fTObject;
  /** Class Name of the block */
```

```
  TString        fClassName;

  // -- Data Specifications --
  /** HLT DataType */
  TString fDataType;
  /** Detector Name, e.g. PHOS */
  TString fDetector;
  /** HLT Specification */
  ULong_t fSpecification;

  // -- Sub Detector Specifications --
  /** SubDetector Name e.g. MODULE */
  Int_t   fSubDetector;
  /** SubSubDetector Name e.g. PARTITION */
  Int_t   fSubSubDetector;
  /** States id block has a subdetector range */
  Bool_t  fHasSubDetectorRange;
  /** States id block has a subsubdetector range */
  Bool_t  fHasSubSubDetectorRange;

  ClassDef( AliHLTHOMERBlockDesc, 2 )
};
#endif
```

Listing H.2: The definition of the class `AliHLTHOMERBlockDesc` to be found in `HLT/BASE/AliHLTHOMERBlockDesc.h` (Formatting adapted to fit in this work).

H.3 HOMER Proxy Handler

```
#ifndef ALIHLTHOMERPROXYHANDLER_H
#define ALIHLTHOMERPROXYHANDLER_H

/* This file is property of and copyright by the ALICE HLT Project
 * ALICE Experiment at CERN, All rights reserved.
 * See cxx source for full Copyright notice  */

/** @file   AliHLTHOMERProxyHandler.h
    @author Jochen Thaeder
    @brief  HOMER proxy handler for HomerManger */

#include "TString.h"
#include "TList.h"
#include "TXMLNode.h"
#include "AliHLTHOMERSourceDesc.h"
#include "AliHLTLoggingVariadicFree.h"

/**
 * @class AliHLTHOMERProxyHandler
 * This Class should handle the communication with the proxy
 * and fill the source list.
 *
 * @ingroup alihlt_homer */
```

```
class AliHLTHOMERProxyHandler : public TObject, public AliHLTLogging
{
public:

  /** constructor */
  AliHLTHOMERProxyHandler();
  /** destructor */
  virtual ~AliHLTHOMERProxyHandler();

  /** Initialize
   *  @return 0 on success, <0 for failure  */
  Int_t Initialize();
  /** Fill's source list, with entries
   *  @return 0 on success, <0 for failure, 1 for no active service  */
  Int_t FillSourceList(TList *srcList);

private:

  /** copy constructor prohibited */
  AliHLTHOMERProxyHandler(const AliHLTHOMERProxyHandler&);
  /** assignment operator prohibited */
  AliHLTHOMERProxyHandler& operator=(const AliHLTHOMERProxyHandler&);

  /** Realms */
  enum HOMERRealms_t {
    kHLT,             /**< HLT realm */
    kACR,             /**< ACR realm */
    kGPN,             /**< GPN realm */
    kKIP,             /**< KIP realm */
    kLoc,             /**< Local realm. needs ssh tunnel */
    kHOMERRealmsMax   /**< Number of enum entries */
  };

  /** Array of proxy nodes per realm */
  static const Char_t *fgkHOMERProxyNode[];

  /** Indentifies the realm and sets it
   *  @return 0 on success, <0 for failure   */
  void IdentifyRealm();
  /** Get xmlrpc response from the proxy
   *  @return 0 on success, <0 for failure   */
  Int_t RequestXmlRpcResponse();
  /** process xmlrpc response and fill the source list
   *  @return 0 on success, <0 for failure, 1 for no active service   */
  Int_t ProcessXmlRpcResponse();
  /** Add a new Service to list
   *  @param xmlNode   Ptr to service node
   *  @return          0 on sucess, <0 for failure   */
  Int_t AddService(TXMLNode *innerNode);

  /** Realm, which can be ACR, GPN, HLT, KIP */
  Int_t       fRealm;
```

```
  /** xmlRPC response */
  TString     fXmlRpcResponse;
  /** List to HOMER sources */
  TList*      fSourceList;                          //! transient

  ClassDef(AliHLTHOMERProxyHandler, 0);
};
#endif
```

Listing H.3: The definition of the class `AliHLTHOMERProxyHandler` to be found in `HLT/BASE/AliHLTHOMERProxyHandler.h` (Formatting adapted to fit in this work).

H.4 HOMER Manager

```
#ifndef ALIHLTHOMERMANAGER_H
#define ALIHLTHOMERMANAGER_H

/* This file is property of and copyright by the ALICE HLT Project
 * ALICE Experiment at CERN, All rights reserved.
 * See cxx source for full Copyright notice  */

/** @file   AliHLTHOMERManager.h
    @author Jochen Thaeder
    @author Svein Lindal
    @brief  Manager for HOMER in aliroot */

#include "TClonesArray.h"
#include "TString.h"
#include "TList.h"
#include "AliHLTHOMERSourceDesc.h"
#include "AliHLTHOMERBlockDesc.h"
#include "AliHLTHOMERReader.h"
#include "AliHLTHOMERProxyHandler.h"
#include "AliHLTLoggingVariadicFree.h"

#define BUFFERSIZE 15

class AliHLTHOMERLibManager;

/**
 * @class AliHLTHOMERManager
 * This Class should handle the communication
 * from the HLT to aliroot. The HLT sends data via
 * the HOMER interface on several TCP ports of nodes
 * in the CERN GPN and DCS network.
 * All this communication is hidden from the user.
 *
 * @ingroup alihlt_homer */

class AliHLTHOMERManager : public AliHLTLogging
{
public:
```

```
  /** default constructor */
  AliHLTHOMERManager();
  /** destructor */
  virtual ~AliHLTHOMERManager();

  /** Initialize */
  Int_t Initialize();
  /** Create Sources List from HOMER-Proxy */
  virtual Int_t CreateSourcesList();
  /** Set state of a source */
  void SetSourceState( AliHLTHOMERSourceDesc* source, Bool_t state);
  /** Get pointer to source List */
  TList* GetSourceList() { return fSourceList; }
  /** Connect to HOMER sources, of a certain detector. */
  Int_t ConnectHOMER( TString detector="ALL" );
  /** Disconnect from HOMER sources */
  void  DisconnectHOMER();
  /** Reconnect from HOMER sources */
  Int_t ReconnectHOMER( TString detector);
  /** Loads the next Event, after being connected */
  virtual Int_t NextEvent();
  /** Loads the next Cycle, after being connected */
  virtual Int_t NextCycle() { return NextEvent(); }
  /** Get event ID */

  ULong_t GetEventID() { return fEventId; }
  Int_t GetNAvailableEvents() { return fNEventsAvailable;}
  /** Get pointer to last requested BlockList */
  TList* GetBlockList() { return fBlockList; }
  TList* GetAsyncBlockList() { return fAsyncBlockList; }
  /** Navigate backwards in event buffer */
  Int_t  NavigateEventBufferBack();
  /** Navigate forwards in event buffer */
  Int_t  NavigateEventBufferFwd();
  /** Set and get the string used to select triggers */
  void SetTriggerString ( TString triggerString ) {
    fTriggerString = triggerString;
  }
  /** Get TriggerString */
  TString GetTriggerString () { return fTriggerString; }
  void SetBlockOwner(Bool_t owner) { fBlockList->SetOwner(owner); }
  Bool_t GetBlockOwner() const { return fBlockList->IsOwner(); }

protected:

  /** Dynamic loader manager for the HOMER library */
  AliHLTHOMERLibManager* fLibManager; //! transient

  /** Indicates, if a sources have changes,
   *  so that one has to reconnect. */
  Bool_t fStateHasChanged;
  /** Check if connected */
```

```
  Bool_t Connected() const { return fConnected; }

private:

  /** copy constructor prohibited */
  AliHLTHOMERManager(const AliHLTHOMERManager&);
  /** assignment operator prohibited */
  AliHLTHOMERManager& operator=(const AliHLTHOMERManager&);

  // == Connection to HOMER ==
  /** Create a readout list for Hostname and ports */
  void CreateReadoutList( const char** sourceHostnames,
                          UShort_t* sourcePorts, UInt_t &sourceCount,
                          TString detector );
  /** Checks if already connected to HOMER sources */
  Bool_t IsConnected() { return fConnected; }
  /** Create and add Block List to Buffer */
  void AddBlockListToBuffer();
  /** Add bocks to asynchronous BlockList */
  void AddToAsyncBlockList();
  /** Add bocks to synchronous BlockList */
  void AddToBlockList();

  // == Block Handling ==
  /** Get pointer to block list in event buffer */
  TList* GetBlockListEventBuffer( );
  /** Get Number of blocks in current event */
  ULong_t GetNBlks() { return fNBlks; }
  /** Handle Blocks and fill them in event buffer or
   *  asyncronous BlockList */
  Int_t HandleBlocks();
  /** Check is block are from syncronous source */
  Bool_t IsSyncBlocks();
  /** Get pointer to block ndx in current event */
  void* GetBlk( Int_t ndx );
  /** Get pointer to current block in current event */
  void* GetBlk() { return GetBlk(fCurrentBlk); }
  /** Get first block in current event */
  void* GetFirstBlk() { fCurrentBlk=0; return GetBlk(0); }
  /** Get next block in current event */
  void* GetNextBlk() { return GetBlk(++fCurrentBlk); }
  /** Get size of block ndx */
  ULong_t GetBlkSize( Int_t ndx );
  /** Get size of current block */
  ULong_t GetBlkSize() { return GetBlkSize( fCurrentBlk ); }
  /** Get origin of block ndx  */
  TString GetBlkOrigin( Int_t ndx );
  /** Get origin of current block */
  TString GetBlkOrigin(){ return GetBlkOrigin( fCurrentBlk ); }
  /** Get type of block ndx */
  TString GetBlkType( Int_t ndx );
  /** Get type of current block */
  TString GetBlkType() { return GetBlkType( fCurrentBlk ); }
```

```
145   /** Get specification of block at ndx in bufferindex */
      ULong_t GetBlkSpecification( Int_t ndx );
      /** Get specification of current block */
      ULong_t GetBlkSpecification() {
        return GetBlkSpecification( fCurrentBlk );
150   }
      /** Check if requested in eve */
      Bool_t CheckIfRequested( AliHLTHOMERBlockDesc* block );
      /** Check trigger decision */
      Bool_t CheckTriggerDecision();
155
      // == Ptr ==
      /** Proxy Handler to get the list of sources */
      AliHLTHOMERProxyHandler * fProxyHandler;  //! transient
      /** Pointer to current HOMER reader */
160   AliHLTHOMERReader* fCurrentReader;        //! transient
      /** List to pointer of HOMER readers */
      TList* fReaderList;

      // == sources ==
165   /** List to HOMER sources */
      TList* fSourceList;
      /** Number of blockes in current event */
      ULong_t fNBlks;
      /** Buffer of EventID of current event */
170   ULong64_t fEventID[BUFFERSIZE];
      /** EventID of current event */
      ULong64_t fEventId;
      /** Current block in current event */
      ULong_t fCurrentBlk;
175   /** List containing asychronous blocks */
      TList* fAsyncBlockList;
      /** List containing sychronous blocks */
      TList* fBlockList;

180   // == event buffer ==
      /** Event Buffer */
      TClonesArray* fEventBuffer;
      /** Buffer index to last received event */
      Int_t  fBufferTopIdx;
185   /** Buffer index to last received event */
      Int_t  fBufferLowIdx;
      /** Buffer index to current event */
      Int_t  fCurrentBufferIdx;
      /** Navigate index through event buffer */
190   Int_t  fNavigateBufferIdx;
      /** Number of available events */
      Int_t  fNEventsAvailable;

      // == status ==
195   /** Shows connection status */
      Bool_t fConnected;
      /** String indicating which trigger should
```

```
     *  be used to select events */
    TString fTriggerString;
    /** Number Events not triggered,
     *  before next triggered event is found */
    Int_t  fNEventsNotTriggered;

    /** Retry reading next event */
    Bool_t fRetryNextEvent;
    /** Is Block owner */
    Bool_t fIsBlockOwner;

    ClassDef(AliHLTHOMERManager, 1);
};
#endif
```

Listing H.4: The definition of the class `AliHLTHOMERManager` to be found in `HLT/BASE/AliHLTHOMERManager.h` (Formatting adapted to fit in this work).

Glossary

AA	nucleus-nucleus
ACORDE	ALICE COsmic Ray DEtector
ACR	ALICE Control Room
ACT	ALICE Configuration Tool
AFS	Andrew File System
ALICE	A Large Ion Collider Experiment
AliRoot	ALIce Root
AMORE	Automatic MOnitoRing Environment
ATLAS	A Toroidal LHC ApparatuS
BMC	Baseboard Management Controller
CASTOR	CERN Advanced STORage manager
CDH	Common Data Header
CERN	European Organization for Nuclear Research
CF	ClusterFinder
CHARM	Computer-Health-And-Remote-Monitoring
CMS	Compact Muon Solenoid
CPU	Central Processing Unit
CR2	Counting Room 2
CR3	Counting Room 3
CTP	Central Trigger Processor
D-RORC	DAQ Read-Out Receiver Card
DAQ	Data AcQuisition
DCA	Distance of Closest Approach
DCS	Detector Control System
DDL	Detector Data Link
DHCP	Dynamic Host Configuration Protocol
DIU	Destination Interface Unit
DNS	Domain Name System
DQM	Data Quality Monitoring
ECS	Experiment Control System

EMCAL ElectroMagnetic CALorimeter
ESD Event Summary Data - `AliESDEvent`

FAIR Facility for Antiproton and Ion Research
FEE Front-End Electronics
FEP Front-End Processor
FMD Forward Multiplicity Detector
FPGA Field Programmable Gate Array
FXS File eXchange Server

GDC Global Data Concentrator
GPN General Purpose Network
GPU Graphics Processing Unit
GUI Graphical User Interface

H-RORC HLT Read-Out Receiver Card
HCDB HLT Conditions DataBase
HLT High-Level Trigger
HMPID High-Momentum Particle Identification Detector
HOMER HLT On-line Monitoring Environment including ROOT

IB InfiniBand
IP Interaction Point
IP Internet Protocol
ITS Inner Tracking System

L2a Level 2 accept
L2r Level 2 reject
LDAP Lightweight Directory Access Protocol
LDC Local Data Concentrator
LEP Large Electron Positron collider
LHC Large Hadron Collider
LHCb LHC beauty
LTS Long Term Support

MRPCs Multi-gap Resistive Plate Chambers

NFS Network File System
NLO Next-to-Leading-Order

OCDB Off-line Conditions DataBase

pA	proton-nucleus
Pb–Pb	lead-lead
PC	Personal Computer
PCI-X	Peripheral Component Interconnect-eXtended
PCIe	PCI express
PHOS	PHOton Spectrometer
PMD	Photon Multiplicity Detector
pp	proton-proton
pQCD	perturbative QCD
QA	Quality Assurance
QCD	Quantum ChromoDynamics
QED	Quantum ElectroDynamics
QFT	Quantum Field Theory
QGP	Quark-Gluon Plasma
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RHIC	Relativistic Heavy Ion Collider
ROI	Region-Of-Interest
RORC	Read-Out Receiver Card
SATA	Serial Advanced Technology Attachment
SCC2	SimpleChainConfig2
SDD	Silicon Drift Detector
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SIU	Source Interface Unit
SPD	Silicon Pixel Detector
SPS	Super Proton Synchrotron
SSD	Silicon Strip Detector
SSH	Secure SHell
SysMES	System Management for networked Embedded Systems and clusters
T-HCDB	TAXI-HCDB
T0	T0 detector
TCP	Transmission Control Protocol
TDS	TCPDumpSubscriber
TOF	Time-Of-Flight detector
TPC	Time Projection Chamber

TR TRacker
TRD Transition Radiation Detector

U rack Unit – 44.45 mm
UPS Uninterruptible Power Supply

V0 V0 detector

XML eXtensible Markup Language

ZDC Zero Degree Calorimeters

List of Figures

1 The HLT crew . . . viii

2.1 QCD string breaking . . . 4
2.2 Hard scattering event . . . 6
2.3 QCD phase diagram . . . 7
2.4 Charged particle pseudo-rapidity density per participant pair . . . 8
2.5 Integrated elliptic flow . . . 10
2.6 Suppression of charged particles, R_{AA} . . . 11
2.7 Jet suppression . . . 11

3.1 Overview of the LHC . . . 14
3.2 Schematic of the ALICE detector . . . 15
3.3 Overview of the ALICE data flow . . . 18
3.4 Overview of the ALICE control systems . . . 21

4.1 Overview of the HLT layers . . . 24
4.2 Overview of the HLT reconstruction hierarchy . . . 25
4.3 Communication between HLT components . . . 27
4.4 Single Instruction Multiple Data approach . . . 29

5.1 First data from the TPC . . . 39
5.2 HLT cluster installation 2007 . . . 40
5.3 Cosmic particle in the TPC 2008 . . . 41
5.4 Cosmic particles in the TPC and ITS 2009 . . . 42
5.5 PHOS in a LHC injection test . . . 42
5.6 On-line reconstructed vertex of run 101498 . . . 43
5.7 One of the first pp collisions at $\sqrt{s} = 900$ GeV . . . 44
5.8 Neutral particles from secondary vertex decays . . . 44
5.9 One of the first pp collisions at $\sqrt{s} = 7$ TeV . . . 45
5.10 Layout of Counting Room 2 . . . 46
5.11 Layout of Counting Room 3 . . . 47
5.12 Layout of the network racks Y2 – Y5 . . . 48
5.13 GigaBit-Ethernet network layout . . . 49
5.14 BackBone InfiniBand network layout . . . 50
5.15 HLT interfaces . . . 59
5.16 HLT–ECS communication . . . 60
5.17 InfoBrowser . . . 70
5.18 SysMES GUI . . . 70
5.19 ESMP2 GUI . . . 72
5.20 HOMER format . . . 76

5.21 HOMER interface 77
5.22 Overview HOMER Manager 78

6.1 Integrated number of pp interactions until September 2010 84
6.2 TPC reconstruction efficiency of run 125633 (LHC10f6a) 85
6.3 TPC reconstruction efficiency (findable) of run 125633 (LHC10f6a) . . . 86
6.4 p_t resolution of run 125633 (LHC10f6a) 87
6.5 TPC reconstruction QA of run 125633 (LHC10f6a) 88
6.6 Trigger efficiency 90
6.7 Trigger purity 91
6.8 p_t distribution of different trigger p_t's 91
6.9 p_t efficiency of different trigger p_t's 92
6.10 Reduction factor 92

D.1 A CN node of the second generation 110

E.1 HLT cluster: user access 113

F.1 ECS states of the HLT 118
F.2 TaskManager states of the HLT 120

List of Tables

6.1 Track selection 87
6.2 Trigger thresholds 89
6.3 Trigger summary results 91

C.1 Optical fiber positions - CR2 Row X - DAQ-to-HLT 105
C.2 Optical fiber positions - CR2 Row Y - DAQ-to-HLT 106
C.3 Optical fiber positions - CR2 Row Z - DAQ-to-HLT 106
C.4 Optical fiber positions - CR2 Raw Z - HLT-to-DAQ 106

D.1 HLT cluster: front-end processor nodes 108
D.2 HLT cluster: front-end processors 108
D.3 HLT cluster: computing nodes 109
D.4 HLT cluster: mass storage & database 110
D.5 HLT cluster: gateway & portal 111
D.6 HLT cluster: development 111
D.7 HLT cluster: monitoring & system management 112
D.8 HLT cluster: switches 112
D.9 HLT cluster: external connections 112

E.1 HLT cluster: access ports 114
E.2 HLT cluster: operator users 114
E.3 HLT cluster: event display portal 115

Bibliography

[1] J. L. Rosner. The Standard Model in 2001. 2001. [arXiv] 0108195v6 [hep-ph].

[2] D. H. Perkins. *Introduction to High Energy physics*. Addisson-Wesley, Fourth edition, 2000. ISBN 0-521-62196-8.

[3] D. J. Griffiths. *Introduction to Elementary Particles*. John Wiley & Sons Inc., 1987. ISBN 0-471-60386-4.

[4] P. W. Higgs. Broken Symmetries and the Masses of Gauge Bosons. *Phys. Rev. Lett.*, 13(16):508–509, 1964. [doi] 10.1103/PhysRevLett.13.508.

[5] F. Abe et al. The CDF Collaboration. Observation of Top Quark Production in $p\bar{p}$ Collisions with the Collider Detector at Fermilab. *Phys. Rev. Lett.*, 74(14):2626–2631, 1995. [doi] 10.1103/PhysRevLett.74.2626.

[6] J. M. Campbell, J. W. Huston, and W. J. Stirling. Hard interactions of quarks and gluons: a primer for LHC physics. *Rep. Prog. Phys.*, 70(1):89–193, 2007. [doi] 10.1088/0034-4885/70/1/R02.

[7] C. A. Loizides. *Jet physics in ALICE*. PhD thesis, Frankfurt, U., 2005. [arXiv] 0501017 [nucl-ex].

[8] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch, and M. Tönnesmann. Jets in hadron-hadron collisions. *Prog. Part. Nucl. Phys.*, 60(2):484–551, 2008. [doi] 10.1016/j.ppnp.2007.12.002.

[9] G. Salam. Towards jetography. *Eur. Phys. J. C*, 67(3):637–686, 2010. [doi] 10.1140/epjc/s10052-010-1314-6.

[10] P. Braun-Munzinger and J. Stachel. The quest for the quark-gluon plasma. *Nature*, 448:302–309, 2007. [doi] 10.1038/nature06080.

[11] E. Shuryak. Physics of strongly coupled quark-gluon plasma. *Prog. Part. Nucl. Phys.*, 62(1):48–101, 2009. [doi] 10.1016/j.ppnp.2008.09.001.

[12] L. McLerran. The physics of the quark-gluon plasma. *Rev. Mod. Phys.*, 58(4):1021–1064, 1986. [doi] 10.1103/RevModPhys.58.1021.

[13] E. V. Shuryak. Quark-Gluon Plasma and hadronic Production of Leptons, Photons and Psions. *Phys. Lett. B*, 78(1):150–153, 1978. [doi] 10.1016/0370-2693(78)90370-2.

[14] H. Satz. The Transition from Hadron Matter to Quark-Gluon Plasma. *Ann. Rev. Nucl. Part. Sci.*, 35:245–270, 1985. [doi] 10.1146/annurev.ns.35.120185.001333.

[15] K. Ackermann et al. The STAR Collaboration. Elliptic Flow in $Au+Au$ Collisions at $\sqrt{s_{NN}}$ = 130 GeV. *Phys. Rev. Lett.*, 86(3):402–407, 2001. [doi] 10.1103/PhysRevLett.86.402.

[16] K. Aamodt et al. The ALICE Collaboration. Elliptic Flow of Charged Particles in Pb–Pb Collisions at $\sqrt{s_{NN}}$ = 2.76 TeV. *Phys. Rev. Lett.*, 105(25):252302, 2010. [doi] 10.1103/PhysRevLett.105.252302.

[17] The CBM Collaboration. The CBM Experiment - Introduction, 2012. [Online] http://www.gsi.de/forschung/fair_experiments/CBM/1intro_e.html.

[18] R. Snellings. The ALICE Collaboration. Heavy-Ion Physics at the LHC with ALICE. In *Proceedings of 34th SLAC Summer Institute on Particle Physics (SSI 2006)*, 2006. [Online] http://www.slac.stanford.edu/econf/C060717/papers/L010.PDF.

[19] J. D. Bjorken. Highly relativistic nucleus-nucleus collisions: The central rapidity region. *Phys. Rev. D*, 27(1):140–151, 1983. [doi] 10.1103/PhysRevD.27.140.

[20] K. Aamodt et al. The ALICE Collaboration. Charged-Particle Multiplicity Density at Midrapidity in Central Pb–Pb Collisions at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$. *Phys. Rev. Lett.*, 105(25):252301, 2010. [doi] 10.1103/PhysRevLett.105.252301.

[21] R. Vogt. *Ultrarelativistic Heavy-Ion Collisions*. Elsevier, 2007. ISBN 978-0-444-52196-5.

[22] S. Voloshin and Y. Zhang. Flow study in relativistic nuclear collisions by Fourier expansion of azimuthal particle distributions. *Z. Phys. C*, 70(4):665–671, 1996. [doi] 10.1007/s002880050141.

[23] K. J. Eskola, H. Honkanen, C. A. Salgado, and U. A. Wiedemann. The fragility of high-pT hadron spectra as a hard probe. *Nucl. Phys. A*, 747(2-4):511–529, 2005. [doi] 10.1016/j.nuclphysa.2004.09.070.

[24] K. Aamodt et al. The ALICE Collaboration. Suppression of charged particle production at large transverse momentum in central Pb–Pb collisions at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$. *Phys. Lett. B*, 696(1-2):30–39, 2011. [doi] 10.1016/j.physletb.2010.12.020.

[25] S. Chatrchyan et al. The CMS Collaboration. Observation and studies of jet quenching in PbPb collisions at $\sqrt{s_{NN}} = 2.76$ TeV. *Phys. Rev. C*, 84(2):024906, 2011. [doi] 10.1103/PhysRevC.84.024906.

[26] M. L. Miller, K. Reygers, S. J. Sanders, and P. Steinberg. Glauber Modeling in High-Energy Nuclear Collisions. *Ann. Rev. Nucl. Part. Sci.*, 57(1):205–243, 2007. [doi] 10.1146/annurev.nucl.57.090506.123020.

[27] T. Renk, H. Holopainen, R. Paatelainen, and K. J. Eskola. Systematics of the charged-hadron P_T spectrum and the nuclear suppression factor in heavy-ion collisions from $\sqrt{s_{NN}} = 200$ GeV to $\sqrt{s_{NN}} = 2.76$ TeV. *Phys. Rev. C*, 84(1):014906, 2011. [doi] 10.1103/PhysRevC.84.014906.

[28] B. Z. Kopeliovich, I. K. Potashnikova, and I. Schmidt. Color transparency and suppression of high-pT hadrons in nuclear collisions. *Phys. Rev. C*, 83(2):021901, 2011. [doi] 10.1103/PhysRevC.83.021901.

[29] S. S. Adler et al. The STAR Collaboration. Absence of suppression in particle production at large transverse momentum in s(NN)**(1/2) = 200-GeV d + Au collisions. *Phys. Rev. Lett.*, 91(7):072303, 2003. [doi] 10.1103/PhysRevLett.91.072303.

[30] J. Adams et al. The PHENIX Collaboration. Evidence from d + Au measurements for final-state suppression of high p(T) hadrons in Au + Au collisions at RHIC. *Phys. Rev. Lett.*, 91(7):072304, 2003. [doi] 10.1103/PhysRevLett.91.072304.

[31] G. Aad et al. The ATLAS Collaboration. Observation of a Centrality-Dependent Dijet Asymmetry in Lead-Lead Collisions at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$ with the ATLAS Detector at the LHC. *Phys. Rev. Lett.*, 105(25):252303, 2010. [doi] 10.1103/PhysRevLett.105.252303.

[32] L. Evans (ed.) and P. Bryant (ed.). LHC Machine. *JINST*, 3:S08001, 2008. [doi] 10.1088/1748-0221/3/08/S08001.

[33] A. A. Alves et al. The LHCb Collaboration. The LHCb Detector at the LHC. *JINST*, 3:S08005, 2008. [doi] 10.1088/1748-0221/3/08/S08005.

[34] G. Aad et al. The ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008. [doi] 10.1088/1748-0221/3/08/S08003.

[35] R. Adolphi et al. The CMS Collaboration. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008. [doi] 10.1088/1748-0221/3/08/S08004.

[36] The ALICE Collaboration. ALICE – Technical Proposal for A Large Ion Collider Experiment at CERN LHC. (CERN/LHCC 1995-71), 1995. ISBN 92-9083-077-8.

[37] K. Aamodt et al. The ALICE Collaboration. The ALICE experiment at the CERN LHC. *JINST*, 3:S08002, 2008. [doi] 10.1088/1748-0221/3/08/S08002.

[38] F. Carminati et al. The ALICE Collaboration. ALICE: Physics Performance Report, Volume I. *J. Phys. G: Nucl. Part. Phys.*, 30(11):1517–1763, 2004. [doi] 10.1088/0954-3899/30/11/001.

[39] B. Alessandro et al. The ALICE Collaboration. ALICE: Physics Performance Report, Volume II. *J. Phys. G: Nucl. Part. Phys.*, 32(10):1295–2040, 2006. [doi] 10.1088/0954-3899/32/10/001.

[40] J.-L. Caron. Layout of the LEP tunnel including future LHC infrastructures, 1997. [Online] http://cdsweb.cern.ch/record/841560.

[41] N. Armesto, (ed.) et al. Heavy Ion Collisions at the LHC - Last Call for Predictions. *J. Phys. G: Nucl. Part. Phys.*, 35(5):054001, 2008. [doi] 10.1088/0954-3899/35/5/054001.

[42] K. J. Eskola, P. V. Ruuskanen, S. S. Räsänen and K. Tuominen. Multiplicities and transverse energies in central AA collisions at RHIC and LHC from pQCD, saturation and hydrodynamics. *Nucl. Phys. A*, 696(3-4):715–728, 2001. [doi] 10.1016/S0375-9474(01)01207-6.

[43] The ALICE Collaboration. ALICE Technical Design Report of the Inner Tracking System (ITS). (ALICE TDR 4, CERN/LHCC 1999-012), 1999. ISBN 92-9083-144-8.

[44] The ALICE Collaboration. ALICE Technical Design Report of the Time Projections Chamber. (ALICE TDR 7, CERN/LHCC 2000-001), 1999. ISBN 92-9083-155-3.

[45] J. Alme et al. The ALICE TPC, a large 3-dimensional tracking device with fast readout for ultra-high multiplicity events. *Nucl. Inst. and Meth. A*, 622(1):316–367, 2010. [doi] 10.1016/j.nima.2010.04.042.

[46] H. Bethe. Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie. *Ann. Phys.*, 397(3):325–400, 1930. [doi] 10.1002/andp.19303970303.

[47] The ALICE DAQ group. ALICE Electronic Logbook. *Website*, 2012. [Online] https://alice-logbook.cern.ch/logbook.

[48] F. Carena et al. The ALICE Electronic Logbook. In *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pages 1–5, 2010. [doi] 10.1109/RTC.2010.5750476.

[49] The ALICE Collaboration. ALICE Technical Design Report of the Transition Radiation Detector (TRD). (ALICE TDR 9, CERN/LHCC 2001-021), 2001. ISBN 92-9083-184-7.

[50] The ALICE Collaboration. ALICE Technical Design Report of the Time-Of-Flight system (TOF). (ALICE TDR 8, CERN/LHCC-2000-012), 2000. ISBN 92-9083-159-6.

[51] The ALICE Collaboration. ALICE Technical Design Report of the Time-Of-Flight system (TOF) Addendum. (Addendum to ALICE TDR 8, CERN/LHCC-2002-016), 2002. ISBN 92-9083-192-8.

[52] The ALICE Collaboration. ALICE Technical Design Report of the high-momentum particle identification (HMPID). (ALICE TDR 1, CERN/LHCC 1998-019), 1998. ISBN 92-9083-134-0.

[53] The ALICE Collaboration. ALICE Technical Design Report of the Photon Spectrometer (PHOS). (ALICE TDR 2, CERN/LHCC 1999-004), 1999. ISBN 92-9083-138-3.

[54] The ALICE Collaboration. ALICE Technical Design Report of the Electromagnetic Calorimeter. (ALICE TDR 14, CERN/LHCC 2008-014), 2008.

[55] The ALICE Collaboration. ALICE Technical Design Report of the Dimuon Forward Spectrometer. (ALICE TDR 5, CERN/LHCC 1999-022), 1999. ISBN 92-9083-148-0.

[56] The ALICE Collaboration. ALICE Technical Design Report of the Forward Detectors: FMD, TO and VO. (ALICE TDR 11, CERN/LHCC 2004-025), 2004.

[57] The ALICE Collaboration. ALICE Technical Design Report of the Photon Multiplicity Detector (PMD). (ALICE TDR 6, CERN/LHCC 1999-032), 1999. ISBN 92-9083-153-7.

[58] The ALICE Collaboration. ALICE Technical Design Report of the Zero-Degree Calorimeter (ZDC). (ALICE TDR 3, CERN/LHCC 1999-005), 1999. ISBN 92-9083-139-1.

[59] LHC Programme Coordination. LHC Luminosity Plots for the 2010 Proton Run. *Website*, 2010. [Online] http://lpc.web.cern.ch/lpc/lumiplots_2010.htm.

[60] LHC Programme Coordination. LHC Luminosity Plots for the 2011 Proton Run. *Website*, 2011. [Online] http://lpc.web.cern.ch/lpc/lumiplots.htm.

[61] LHC Programme Coordination. LHC Luminosity Plots for the 2010 Heavy-Ion Run. *Website*, 2010. [Online] http://lpc.web.cern.ch/lpc/lumiplots_ions_2010.htm.

[62] LHC Programme Coordination. LHC Luminosity Plots for the 2011 Heavy-Ion Run. *Website*, 2011. [Online] http://lpc.web.cern.ch/lpc/lumiplots_ion.htm.

[63] The ALICE Collaboration. ALICE Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger, and Control System. (ALICE TDR 10, CERN/LHCC 2003-062), 2004. ISBN 92-9083-217.

[64] R. Divià, P. Jovanovic, and P. Vande Vyvre. Data Format over the ALICE DDL. ALICE-INT-2002-010, ver. 11, 2007. [Online] https://edms.cern.ch/document/340186.

[65] C. Cheshkov, R. Divià, and T. M. Steinbeck. Identification of DDL links in ALICE data. ALICE-INT-2007-016, ver. 3, 2008. [Online] https://edms.cern.ch/document/871996.

[66] S. Boettger, T. Breitner, U. Kebschull, C. Lara, J. Ulrich, and P. Zelnicek. Autonomous System Management for the ALICE High-Level-Trigger Cluster using the SysMES framework. *J. Phys.: Conf. Ser.*, 331(5):052003, 2011. [doi] 10.1088/1742-6596/331/5/052003.

[67] T. Alt and V. Lindenstruth. The ALICE HLT Read-Out Receiver Card. *GSI Sci. Rep.*, page 286, 2005. [Online] GSI Scientific Report 2005.

[68] T. M. Steinbeck. *A Modular and Fault-Tolerant Data Transport Framework*. PhD thesis, Heidelberg, U., 2004. [arXiv] 0404014 [cs].

[69] T. M. Steinbeck et al. An Object-Oriented Network-Transparent Data Transportation Framework. *IEEE Trans. Nucl. Sci.*, 49(2):455–459, 2002. [doi] 10.1109/TNS.2002.1003773.

[70] T. M. Steinbeck et al. New experiences with the ALICE High Level Trigger Data Transport Framework. In *Proc. Computing in High Energy Physics Conf. (CHEP04)*, 2004. [Online] http://chep2004.web.cern.ch/chep2004.

[71] K. Haviland, D. Gray, and B. Salama. *UNIX System Programming*. Addisson-Wesley, Second edition, 1998. ISBN 0-201-87758-9.

[72] R. Divià and T. M. Steinbeck. Data format and specifications for the HLT-to-DAQ interface. ALICE-INT-2007-015, ver. 3, 2008. [Online] https://edms.cern.ch/document/871995.

[73] The ALICE offline group. ALICE Experiment: Offline Project. *Website*, 2012. [Online] http://aliceinfo.cern.ch/Offline/.

[74] P. Hristov. *AliRoot Primer*. The ALICE Collaboration, 2006.

[75] The ALICE offline group. ALICE Experiment: AliROOT SVN Repository. *Website*, 2012. [Online] http://alisoft.cern.ch/viewvc/?root=AliRoot.

[76] M. Richter. *Development and Integration of on-line Data Analysis for the ALICE Experiment*. PhD thesis, Bergen, U., 2009. [Online] http://cdsweb.cern.ch/record/1331812.

[77] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, 21(9):948–960, 1972. [Online] http://portal.acm.org/citation.cfm?id=1952456.1952459.

[78] R. Duncan. A Survey of Parallel Computer Architectures. *Computer*, 23(2):5–16, 1990. [doi] 10.1109/2.44900.

[79] S. Gorbunov et al. ALICE HLT high speed tracking and vertexing. In *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pages 1–4, 2010. [doi] 10.1109/RTC.2010.5750344.

[80] S. Gorbunov et al. ALICE HLT High Speed Tracking on GPU. *IEEE Trans. Nucl. Sci.*, 58(4). [doi] 10.1109/TNS.2011.2157702.

[81] T. Alt. *FPGA Clusterfinder Implementation*. PhD thesis, Frankfurt, U., 2012. To be submitted.

[82] A. S. Vestbø. *Pattern Recognition and Data Compression for the ALICE High Level Trigger*. PhD thesis, Bergen, U., 2004. [arXiv] 0406003 [physics].

[83] J. Berger et al. TPC data compression. *Nucl. Inst. Meth. A*, 489(1-3):406–421, 2002. [doi] 10.1016/S0168-9002(02)00792-1.

[84] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proc. IRE*, 40(9):1098–1101, 1952. [doi] 10.1109/JRPROC.1952.273898.

[85] N. Nikityuk and V. Samoilov. Review of the trigger systems of the ATLAS and CMS detectors at the LHC. *Physics of Particles and Nuclei*, 38(5):659–697, 2007. [doi] 10.1134/S106377960705005X.

[86] V. Lindenstruth and I. Kisel. Overview of trigger systems. *Nucl. Instr. Meth. A*, 535(1-2):48–56, 2004. [doi] 10.1016/j.nima.2004.07.267.

[87] J. Thäder. *ALICE HLT TPC Experiment and Analysis*. Diploma thesis, Heidelberg, U., 2006.

[88] P. Lebrun. Interim Summary Report on the Analysis of the 19 September 2008 Incident at the LHC, 2008. [Online] https://edms.cern.ch/document/973073/1.

[89] K. Aamodt et al. The ALICE Collaboration. First proton–proton collisions at the LHC as observed with the ALICE detector: measurement of the charged particle pseudorapidity density at sqrt(s) = 900 GeV. *Eur. Phys. J. C*, 65(1-2):111–125, 2010. [doi] 10.1140/epjc/s10052-009-1227-4.

[90] J. Podolanski and R. Armenteros. Analysis of V-events. *Phil. Mag. Series 7*, 45(360):13–30, 1954. [doi] 10.1080/14786440108520416.

[91] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. Beowulf: A Parallel Workstation For Scientific Computation. In *Proceedings, 24th International Conference on Parallel Processing*, pages 11–14, 1995.

[92] D. Ridge, D. Becker, P. Merkey, and T. Sterling. Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs. In *Proceedings, IEEE Aerospace*, pages 79–91, 1997.

[93] R. G. Brown. *Engineering a Beowulf-Style Compute Cluster*. Online, 2009. [Online] http://www.phy.duke.eduu/∼rgb/Beowulf/beowulf_book.php.

[94] P. Calleja. 36 Gigaflops to 18 Teraflops in 30 Day's. *Talk*, 2007. [Online] http://www.cse.scitech.ac.uk/disco/mew17/talks/Calleja_Nortel_MEW17.pdf.

[95] P. Calleja (University of Cambridge, Centre for Scientific Computing), 2008. Private Communication.

[96] The Server Rack FAQ. Define : EIA-310. *Website*, 2012. [Online] http://www.server-racks.com/eia-310.html.

[97] The Server Rack FAQ. Define : Rack Unit "U" or "RU". *Website*, 2012. [Online] http://www.server-racks.com/rack-unit-u-ru.html.

[98] Information Sciences Institute, University of Southern California. RCF 791 - INTERNET PROTOCOL. *Website*, 1981. [Online] http://tools.ietf.org/html/rfc791.

[99] L. L. Peterson and B. S. Davie. *Computer Networks, A systems approach*. Morgan Kaufmann Publishers, Second edition, 2000. ISBN 1-55860-577-0.

[100] R. E. Panse. *CHARM-Card: Hardware Based Cluster Control And Management System*. PhD thesis, Heidelberg, U., 2009.

[101] S. Kalcher (University of Frankfurt, FIAS), 2009. Private Communication.

[102] InfiniBand Trade Association. About InfiniBand™. *Website*, 2012. [Online] http://www.infinibandta.org/content/pages.php?pg=about_us_infiniband.

[103] J. P. R. Middelink. Bigphysarea Kernel patch. *Website*, 2003. [Online] http://www.polyware.nl/∼middelink/En/hob-v4l.html#bigphysarea.

[104] Adaptec by PMC. Which RAID Level is Right for Me? *Website*, 2012. [Online] http://www.adaptec.com/en-us/_common/compatibility/_education/raid_level_compar_wp.htm.

[105] MIT Kerberos. Kerberos: The Network Authentication Protocol. *Website*, 2012. [Online] http://web.mit.edu/kerberos/.

[106] Debian. Debian – The Universal Operating System. *Website*, 2012. [Online] http://www.debian.org.

[107] Canonical Ltd. Home — Ubuntu. *Website*, 2012. [Online] http://www.ubuntu.com.

[108] Ø. Haaland. *Autonomic Operation of a Large High Performance Compute Cluster*. PhD thesis, Bergen, U., 2012. To be submitted.

[109] Ø. Haaland (University of Bergen), 2008. Private Communication.

[110] J. H. Howard et al. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, 1988. [doi] 10.1145/35037.35059.

[111] openafs.org. OpenAFS. *Website*, 2012. [Online] http://www.openafs.org.

[112] IT-DSS CERN. CASTOR Home Page — castor.web.cern.ch. *Website*, 2012. [Online] http://castor.web.cern.ch.

[113] R. Brun et al. ROOT: An object oriented data analysis framework. *Nucl. Inst. and Meth. A*, 389(1-2):81–86, 1997. [doi] 10.1016/S0168-9002(97)00048-X.

[114] R. Brun et al. ROOT — A Data Analysis Framework. *Website*, 2012. [Online] http://root.cern.ch.

[115] The GEANT Collaboration. GEANT - Detector Description and Simulation Tool. *Website*, 2003. [Online] http://wwwasd.web.cern.ch/wwwasd/geant.

[116] The GEANT Collaboration. GEANT3 SVN Repository. *Website*, 2012. [Online] http://root.cern.ch/viewvc/?root=geant3.

[117] The AliEn Development Team. Alien. *Website*, 2012. [Online] http://alien2.cern.ch/.

[118] J. L. Furlani. Modules: Providing a flexible user environment. In *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA 1991)*, pages 141–152, 1991. [Online] http://modules.sourceforge.net/docs/Modules-Paper.pdf.

[119] J. L. Furlani and P. W. Osel. Abstract yourself with modules. In *Proceedings of the Tenth Large Installation Systems Administration Conference (LISA 1996)*, pages 193–204, 1996. [Online] http://modules.sourceforge.net/docs/absmod.pdf.

[120] J. L. Furlani and P. W. Osel. Modules – Software Environment Management. *Website*, 2011. [Online] http://modules.sourceforge.net.

[121] S. R. Bablok. *Heterogeneous Distributed Calibration Framework for the High Level Trigger in ALICE*. PhD thesis, Bergen, U., 2008.

[122] T. M. Steinbeck. HLT Online Monitoring Environment (including ROOT). *Website*, 2010. [Online] https://twiki.cern.ch/twiki/bin/viewauth/ALICEHLT/MediaWiki/HOMER.

[123] A. Colla and J. F. Grosse-Oetringhaus. The Shuttle Framework - A system for automatic readout and processing of conditions data. ALICE-INT-2008-011, ver. 01, 2008. [Online] https://edms.cern.ch/document/924807.

[124] T. M. Steinbeck. HLT Control Software (TaskManager). *Website*, 2010. [Online] http://www.kip.uni-heidelberg.de/wiki/HLT/index.php/TaskManager.

[125] V. Altini et al. The ALICE DAQ online databases. In *Real Time Conference (RT), 2009 16th IEEE-NPSS*, pages 361–365, 2009. [doi] 10.1109/RTC.2009.5322155.

[126] P. Zelnicek. *A distributed fault-tolerant agent network for multilevel operation and control interfaces*. PhD thesis, Frankfurt, U., 2012. To be submitted.

[127] The Avahi Team. Avahi. *Website*, 2012. [Online] http://avahi.org.

[128] UserLand Software, Inc. XML-RPC Home Page. *Website*, 2012. [Online] http://www.xmlrpc.com.

[129] A. Telesca et al. The alice data quality monitoring system. In *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pages 1–6, 2010. [doi] 10.1109/RTC.2010.5750364.

[130] OpenGL.org Organization. OpenGL Overview. *Website*, 2012. [Online] http://www.opengl.org/about/overview.

[131] M. Tadel. Raw-data display and visual reconstruction validation in ALICE. *J. Phys.: Conf. Ser.*, 119(3):032036, 2008. [doi] 10.1088/1742-6596/119/3/032036.

[132] M. Tadel. The new generation of OpenGL support in ROOT. *J. Phys.: Conf. Ser.*, 119(4):042028, 2008. [doi] 10.1088/1742-6596/119/4/042028.

[133] K. Schweda. Run Coordination Report. *Talk*, 2010. [Online] https://indico.cern.ch/getFile.py/access?contribId=0&resId=1&materialId=slides&confId=83443.

[134] S. Gorbunov. *On-line reconstruction algorithms for the CBM and ALICE experiments.* PhD thesis, Frankfurt, U., 2012.

[135] T. Sjöstrand. PYTHIA Monte Carlo Model.

[136] Zhongbao Yin. Update on HLT purity and efficiency. *Talk*, 2009. [Online] https://indico.cern.ch/getFile.py/access?contribId=3&resId=1&materialId=slides&confId=67223.

[137] J. Thäder. Status of HLT Mode C triggering. *Talk*, 2009. [Online] https://indico.cern.ch/getFile.py/access?contribId=1&resId=0&materialId=slides&confId=67228.

[138] J. Thäder. The ALICE High-Level Trigger Class Naming Schema. *Talk*, 2009. [Online] https://indico.cern.ch/materialDisplay.py?contribId=2&materialId=slides&confId=67228.

[139] The ALICE HLT Collaboration. HLT Wiki. *Website*, 2010. [Online] http://www.kip.uni-heidelberg.de/wiki/HLT/index.php.

Index

D-RORC, 20, 25
HLT–ECS proxy, 56
HLT operator, 40
H-RORC, 26

ACORDE, 14, 18
ACR, 21, 113
ACT, 21, 38
ALICE, i, 13
AliEn, 57
AliRoot, 28, 56
AliRootWrapperSubscriber, 26
ATLAS, 13, 14, 31

Baryon, 3

CERN, 13
CMS, 13, 14, 17, 31
CN, 109
Coulomb potential, 4
CR2, 45
CR3, 45
CTP, 18, 19

DAQ, v, 18, 20, 24
    AMORE, 81
    LDC, 25
DCS, v, 21, 36
DDL, 20, 24, 25
DIU, 26
DQM, 81

ECS, iv, 21, 24, 36
EIA/ECA-310, 46
EMCAL, 14, 17
ESD, 29

FAIR, 7
FEE, 18
FEP, iv, 26, 50, 107
Flow, 9
flow
    directed flow, 9
    elliptic flow, 9
FMD, 17
FPGA, 26

GDC, 20
Geant3, 56

hadron, 3
HCDB, 30, 61
HLT, ii, 1, 18–20, 23, 127
    CHARM, 107
    CN, 109
    FEP, 107
    RunControl, 62
    RunManager, 63
    TaskManager, 63
HLT Analysis Framework, 26, 28
HLT components, 25
HLT–ECS proxy, 60, 69
HLTOUT, 27
HMPID, 14, 17

InfiniBand, 48
InfoBrowser, 69
IP, 14
ITS, 14, 15

L3 Magnet, 14
LDC, 20, 25
LEP, 3
Lepton, 3
LHC, i, 1, 13
LHCb, 13, 14

MasterTaskManager, 63
Meson, 3
Mode A, 20
Mode B, 20

Mode C, 20

NLO pQCD, 5

OCDB, v, 30, 36, 61

Pb–Pb, iii, 8, 16, 19, 83
PCI-X, 20
PENDOLINO, 56
PHOS, 14, 17
PMD, 17
pp, 5, 13
pQCD, 5

QCD, 3
QED, 4
QGP, 5, 6

RHIC, 3
ROI, 21, 24
ROOT, 56
RunControl, 40, 57, 62
RunManager, 63

SDD, 15
SIMD, 28, 29
SimpleChainConfig2, 64
SISD, 28
SIU, 26
SPD, 15
SPS, 3
SSD, 15
SysMES, 24

T0, 18
TaskManager, 63
TAXI, 56
TDS, v, 28, 36, 73
TI 2, 14
TI 8, 14
TOF, 14, 16
TPC, ii, 14, 16, 23
TRD, 14, 16

V0, 18, 29

ZDC, 18

Jochen Thäder

Havelstrasse 22
D – 64295 Darmstadt
Tel.: +49 177 384 2223
e-mail: jochen@thaeder.de

PERSÖNLICHE DATEN

Geboren am 4. November 1978 in Heidelberg

Staatsangehörigkeit deutsch

BERUFLICHE TÄTIGKEITEN

05/2010 – lfd. *Wissenschaftlicher Mitarbeiter*
Research Division and ExtreMe Matter Institute EMMI, GSI Helmholtz-zentrum für Schwerionenforschung GmbH, Forschung

ALICE Kollaboration

04/2011 – lfd. *Conference Committee Member*
Review und Genehmigung aller Computing als auch Physik bezogenen Abstracts, Poster und Präsentationen

06/2011 – lfd. *TPC Offline Coordinator*
Koordinierung der Software der TPC, insbesondere HLT bezogener Themen

09/2011 *Period Run Coordinator*
Überwachung und Koordinierung der operativen Datennahme und Schichtleiter, Durchführung und Leitung täglicher operativen Besprechungen, Kontaktperson zur operativen LHC Leitung

10/2011 – 12/2011 *Deputy Trigger Coordinator*
Überwachung und Koordinierung der Trigger Operativen in der 2011 Schwerionen Periode des LHC, Erstellung der Trigger Menüs und Statistiken

01/2010 – lfd. *Promotionsstudent*
Universität Frankfurt, Lehrstuhl für Architektur von Hochleistungsrechnern
Wechsel an die Universität Frankfurt durch Umzug des Lehrstuhls bei gleichen Tätigkeiten

Betreuung:
Prof. Dr. H. Appelshäuser, Prof. Dr. V. Lindenstruth

01/2007 – 12/2009 *Stipendium*
Graduiertenkolleg International Research Training Group „Development and Application of Intelligent Detectors“

10/2006 – 12/2009 *Promotionsstudent*
Universität Heidelberg, Lehrstuhl für Technische Informatik
Tätigkeiten: Forschung, Lehre, Projektmanagement

Forschung:

Dissertationsthema:
„Commissioning of the ALICE High-Level Trigger “

Betreuung:
Prof. Dr. V. Lindenstruth, Prof. Dr. D. Röhrich (U. Bergen, Norwegen)

Lehre:

Betreuung von Diplomanden, Praktikanten, Wissenschaftlichen Hilfskräften und CERN Sommerstudenten.

Projektmanagement:

Technischer Koordinator des ALICE High-Level Triggers
Projekt-Plannung, Ausführungs-Planung in einem multi-nationalen Forschungsgrossprojekt, Kontakt zu externen Projektpartnern, Kontakt zu externen Firmen

Cluster Administrator des ALICE High-Level Triggers
Hardware und Software Administration eines Linux basiertem High-Performance Compute Clusters

10/2007 – 04/2008 *Dozent*
Universität Heidelberg
Basiskurs Schlüsselkompetenzen in der Physik

10/2006 – 04/2007 *Dozent*
Universität Heidelberg
Basiskurs Schlüsselkompetenzen in der Physik

04/2002 – 07/2002 *Tutor*
Universität Heidelberg
Übungsgruppenleiter Vorlesung „Technische Informatik“

WISSENSCHAFTLICHE UND SONSTIGE ARBEITSERFAHRUNG

06/2009 *Summer School*
CERN, Genf, Schweiz
Fourth CERN-Fermilab Hadron Collider Physics Summer School

01/2006 – 08/2006 *Hilfskraft*
certon systems GmbH
PC - Hardware Entwicklung

04/2002 *Miniforschung*
Deutsches Elektronen-Synchrotron (DESY), Hamburg
Vergleich von RICH Rekonstruktionsalgorithmen für HERA-B

02/2002 – 04/2002 *Wissenschaftliche Hilfskraft*
Physikalisches Instituts der Universität Heidelberg
Design und Erstellung der Instituts-Webseiten

08/2001 *Sommerstudent*
Paul Scherrer Institut (PSI), Villigen, Schweiz
Monte Carlo Simulation von ultrakalten Neutronen in einer Speicherflasche für UCN

11/2000 – 08/2002 *Wissenschaftliche Hilfskraft*
Interdisziplinären Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg
Administration von Websites und Datenbanken in der Arbeitsgruppe Digitale Bildverarbeitung

STUDIUM

13.10.2006 Abschluss zum Diplom Physiker, Note: gut (1,9)
Diplomarbeit zum Thema „ALICE HLT TPC Experiment and Analysis“ am Lehrstuhl für Technische Informatik, Universität Heidelberg

10/1999 - 10/2006 Universität Heidelberg
Studium der Physik (Diplom)
Schwerpunkte: Experimentelle Teilchenphysik und Technische Informatik

AUSLANDSAUFENTHALTE

01/2007 – 05/2010	CERN, Genf, Schweiz Forschungsaufenthalt
11/2008 - 12/2008	Lawrence Berkeley National Laboratory, Berkeley, USA Forschungsaufenthalt
08/2002 - 06/2003	University of Oklahoma, Norman, USA Graduate College, Physics

VERÖFFENTLICHUNGEN MIT SIGNIFIKANTER BETEILIGUNG

K. Aamodt et al. The ALICE Collaboration. Suppression of charged particle production at large transverse momentum in central Pb–Pb collisions at √ sNN = 2.76 TeV. Phys. Lett. B, 696(1-2):30–39, 2011.

K. Aamodt et al. The ALICE Collaboration. First proton–proton collisions at the LHC as observed with the ALICE detector: measurement of the charged particle pseudorapidity density at sqrt(s) = 900 GeV. Eur. Phys. J. C, 65(1-2):111–125, 2010.

K. Aamodt et al. The ALICE Collaboration. The ALICE experiment at the CERN LHC. JINST, 3:S08002, 2008.

H. Albrecht, W. Gradl, J. Thader, A. Belkov, Research of particle identification models in the HERA-B ring imaging Cherenkov counter. Phys.Part.Nucl.Lett. 1:274-278, 2004

SCHULE UND WEHRDIENST

09/1998 - 09/1999	Wehrdienst, beendet mit Dienstgrad Unteroffizier der Reserve
06/1998	Abitur am Technischen Gymnasium Heidelberg mit der Note 1,4

FREMDSPRACHENKENNTNISSE

Englisch – Verhandlungssicher in Wort und Schrift

Französisch – Grundkenntnisse in Wort und Schrift

Spanisch – Basiskenntnisse in Wort und Schrift

IT KENNTNISSE

Entwicklung	C/C++, BASH, PHP, (D)HTML – sehr gute Kenntnisse
	VHDL, Fortran77, SQL, Assembler – Basiskenntnisse
Anwendungen	AliROOT, ROOT
	MS Office, OpenOffice

SONSTIGES

Seit 11/2004	Mitglied der Deutschen Physikalischen Gesellschaft (DPG)
Seit 1995	Mitglied und ehrenamtliche Aufsicht der Schützengesellschaft von 1781 Neckargemünd e.V.

Darmstadt, den 18. Juni 2012