DYNAMIC INTEGRATION OF CLOUD RESOURCES INTO LOCAL COMPUTING CLUSTERS AND CALIBRATION OF THE JET ENERGY SCALE WITH THE CMS DETECTOR

Master’s Thesis
by
Georg Fleig

Institute of Experimental Nuclear Physics (IEKP)
Karlsruhe Institute of Technology

Advisor: Prof. Dr. Günter Quast
Coadvisor: Prof. Dr. Ulrich Husemann

9. March 2016
DYNAMISCHE INTEGRATION VON CLOUD-RESSOURCEN IN LOKALE RECHENCLUSTER UND KALIBRIERUNG DER JET-ENERGIESKALA DES CMS-DETEKTORS

Masterarbeit
von
Georg Fleig

Institut für Experimentelle Kernphysik (IEKP)
Karlsruher Institut für Technologie

Referent: Prof. Dr. Günter Quast
Korreferent: Prof. Dr. Ulrich Husemann

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.


Georg Fleig
7. Jet Energy Calibration with Z+Jet Events
   7.1. CMS Jet Energy Calibration
       7.1.1. Absolute Residual Corrections with Z(→μμ)+Jet Events
   7.2. Calibrating the Absolute Jet Energy Scale with Z(→μμ)+Jet Events
       7.2.1. The new Calibration Software
       7.2.2. Event and Physics Objects Selection
       7.2.3. Applying Previous Jet Energy Corrections Steps
       7.2.4. Missing Transverse Energy Correction
       7.2.5. Event Weight Calculation
       7.2.6. Specific Selections for the Z+Jet Topology
       7.2.7. Jet Response Determination
   7.3. Comparison of Simulated Events at 8 and 13 TeV
       7.3.1. Object Properties
       7.3.2. Jet Response
   7.4. Summary

8. Conclusion

A. Appendix: ROCED log file

B. Appendix: Jet Energy Corrections
   B.1. Datasets
   B.2. JEC Level Name Mapping
   B.3. Muon and Jet ID
   B.4. Additional Plots of the 8/13 TeV Monte-Carlo Comparison

Bibliography

List of Figures

List of Tables
CHAPTER 1

INTRODUCTION

While in the past, science was driven by individual persons or small groups, it has primarily become a group effort in the last decades. Worldwide collaborations of researchers, enormous machines and complex infrastructure are required to obtain a deeper understanding of Nature.

One of the key theories in physics is the Standard Model of particle physics. Probing and extending it is the collective work of many researchers and the motivation for constructing high energy particle colliders. At present, the Large Hadron Collider (LHC) at CERN is the most powerful particle collider. After a two-year maintenance and upgrade period, the LHC has been in operation again since 2015 with an increased luminosity and a center-of-mass energy of 13 TeV.

While the rise in energy allows experiments, such as the CMS detector, to observe physics processes at previously inaccessible energy ranges, the increased luminosity results in enormous amounts of data. In addition, extensive simulations of these processes are required for comparing the recorded data with theory predictions. This imposes new challenges on data storage and processing. To meet this rising demand for computing resources, new methods of resource acquisition need to be explored. This thesis describes the dynamic utilization of a shared computing cluster made accessible to the high energy physics (HEP) community by using modern virtualization technologies. The concept of this shared cluster, as well as its integration into existing computing infrastructure, is an innovative and efficient way to provide the necessary computing resources for physics analyses.

Furthermore, analyses of physics processes require sophisticated and specialized software. As part of this thesis, large contributions were made to a modern and modular software analyzing $Z(\rightarrow \mu\mu)+\text{jet}$ events to derive corrections for the jet energy scale. These corrections are necessary as particle jets are abundant in LHC proton-proton collisions, and their energy measurement constitutes a dominant uncertainty source in most analyses of the CMS Collaboration.
Chapters 2, 3 and 4 provide the relevant background information for this thesis. Chapter 2 outlines the relevant topics of particle physics and gives an overview of the CMS experiment at the Large Hadron Collider. Chapter 3 introduces conventional computing paradigms and provides a description of common HEP workflows and worldwide distributed computing. Chapter 4 provides an overview of the topic of cloud computing. Additionally, the benefits of cloud computing and virtualization for the HEP community are discussed.

Chapter 5 describes the existing computing infrastructure at the Institute of Experimental Nuclear Physics (IEKP) and the technologies required for dynamic integration of cloud resources. A cloud meta-scheduler is introduced and its extensions with new features to work with the institute’s batch system are detailed. In addition, the creation of virtual HEP worker nodes is described. In Chapter 6, the unique hybrid setup of a high-performance computing cluster located at the University of Freiburg is introduced. Furthermore, the transparent and dynamic embedding of cloud resources at this cluster is described. The usability of these resources is studied by processing various HEP workflows on a similar prototype cluster. Finally, a long-term evaluation of the system’s stability and an outlook for future developments is presented.

Chapter 7 introduces the CMS jet energy calibration procedure, followed by a detailed description of the new calibration software for absolute jet energy scale corrections. This software was developed as part of this thesis, benefiting from a modern, commonly shared codebase. The calibration procedure is illustrated by using the first accumulated data at a center-of-mass energy of 13 TeV. In addition, a comparison of simulated events for center-of-mass energies of 8 TeV and 13 TeV, which was performed during the preparations for LHC Run II, is presented.

Finally, Chapter 8 summarizes the results of this thesis.
Mankind always sought a deeper understanding of itself and its surroundings. Many scientific fields of research evolved out of this drive for more knowledge. One of these is the field of particle physics, also referred to as high energy physics (HEP).

This chapter gives an outline of particle physics and an overview of the CMS experiment at the Large Hadron Collider, being a major experiment in this field. The level of detail is intentionally limited to the scope of this thesis’ topic. More detailed descriptions of the LHC and the CMS experiment can be found in the corresponding technical design reports [1–4].

2.1. Particle Physics Fundamentals

Starting with the periodic table of chemical elements, the idea of having smaller building blocks forming the known atoms grew. Nowadays it is well established that atoms are not the most basic units of matter. The Standard Model is a theory that describes the smallest constituents of all known particles and its interactions. While the Standard Model is able to explain most of the phenomena observed, it does not yet give a complete description of Nature. Most importantly, gravity is not explained by the Standard Model. Theoretical extensions are referred to as physics beyond the Standard Model.

The two main groups of particles are the fermions and the bosons. While the former make up the matter in the universe, the latter are seen as the force carriers of the electromagnetic (photon), weak ($W^\pm$ and $Z$ bosons) and strong force (gluons) and the result of the spontaneous symmetry breaking, the Higgs boson [5]. The fermions are again subdivided into quarks and leptons coming in six flavors each. The quark flavors are up, down, charm, strange, top and bottom. The leptons come in three pairs of a
charged lepton (electron, muon and tauon) and its respective neutral lepton-neutrino (e.g. muon-neutrino), representing three families.

Experiments in particle physics often make use of the mass-energy-equivalence. Particles are brought to collision at high energies to either study their constituents or to create new, heavier particles. In both cases, the bosons of the Standard Model mediate the energy between particles. When protons are brought to collision, their constituting quarks and gluons may leave the proton. Because of color confinement, they can only exist in color-neutral states, which is not given for a single, colored quark. To obey confinement, new quarks are created out of the vacuum and colorless objects, so called hadrons, are formed. A group of such collinear hadrons is called a particle jet with an envelope shape similar to a cone. Jets are a typical signature in particle collisions at the LHC.

2.2. The Large Hadron Collider

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is located beneath the French-Swiss border [6]. It is the world’s most powerful particle accelerator for protons (and heavy ions). It was built inside the former LEP tunnel to further study the Standard Model and possible extensions. While the LHC itself has a circumference of 27 km, several smaller and older pre-accelerators speed up the particles before injecting them into the LHC, as shown in Figure 2.1.

The LHC consists of two beam pipes for particle circulation in opposite directions with 0.999999999% of the speed of light. The particle beams consist of discrete bunches, each holding over $10^{11}$ protons. During the years 2010–2012 (LHC Run I) the separation between two bunches $t_b$ was 50 ns and the center-of-mass energy $\sqrt{s}$ was 7–8 TeV. For Run II, which started in 2015 after a two year upgrade phase, both parameters were optimized to $t_b = 25$ ns and $\sqrt{s} = 13$ TeV in order to operate closer to the original design specifications of the LHC. Both parameters increase the luminosity $L$ which is defined as

$$L = \frac{\dot{N}}{\sigma},$$

where $\dot{N}$ refers to the event rate of a specific physics process and $\sigma$ to the associated cross section [8]. Therefore, the increased luminosity of up to $2 \cdot 10^{34}$ cm$^{-2}$s$^{-1}$ enables the search for physics processes with low cross sections for LHC Run II.

The two particle beams are crossing each other at four interaction points, resulting in many particle collisions being recorded by the experiments. The LHC tunnel houses four major experiments along the accelerator ring. ATLAS [9] and CMS [10] serve as

---

1Large Electron-Positron Collider
multi-purpose experiments, allowing the study of a wide range of physics processes. As both experiments are independent of each other, they can cross-check each other’s results. The most important achievement of both experiments so far is the discovery of the Higgs boson in 2012. The ALICE experiment [11] is designed to study quark-gluon plasmas using heavy ion (Pb) collisions, whereas the LHCb experiment [12] is specialized in bottom quark studies.

2.3. The CMS Detector

The Compact Muon Solenoid (CMS) detector is one of the two multi-purpose detectors at the LHC. It was built to identify a wide range of particles and perform a variety of physics analyses. As the name indicates, the detector is relatively small and heavy (compared to ATLAS) and has a particularly good muon detection. The detector has a cylindrical shape with a length of 28.7 m and a diameter of 15 m and weighs 14 kt. It consists of several subdetectors, each built for a specific task. They are arranged in layers around the interaction point at the center of the detector.
Figure 2.2: A cutaway view of the CMS detector [13]. All subdetectors are organized in layers around the interaction point, together with the endcaps and forward section. The $z$-axis of the CMS coordinate system is orientated anti-clockwise along the beam pipe, while the $x$-axis points to the center of the LHC ring and the $y$-axis upwards. The azimuthal angle $\phi$ is measured from the $x$-axis in the $x$-$y$-plane, while the polar angle $\theta$ is measured starting at the positive $z$-direction. In high energy physics $\theta$ is usually replaced by the Lorentz-invariant pseudorapidity $\eta$.

Detector design

The whole detector, including the most important subdetectors, is shown in Figure 2.2. Closest to the beam pipe and the interaction point is the tracker, consisting of silicon based pixel detector elements at the center and silicon based strip elements for the outer regions. Its purpose is the measurement of particle trajectories and the locations of interaction vertices. Next is the lead tungstate electromagnetic calorimeter (ECAL) for the energy measurement of electrons, positrons and photons. This is followed by the hadronic calorimeter (HCAL), which is a brass sampling calorimeter and measures the energy of hadronic jets. These subdetectors are surrounded by a massive superconducting solenoid with a homogeneous magnetic field of 3.8 T. Charged particles inside the magnetic field are forced on a bent path whose curvature allows to determine the charge sign and momentum of these particles. As muons deposit only very little energy, they are not stopped by the solenoid. Therefore, the muon system is placed outside, paired with the iron return yoke of the solenoid. Hits in the muon system enable a precise energy measurement [10]. The reconstruction of muons and jets is described later in this section.

A physics event

An event is the result of a one-time readout of most of the detector electronics of CMS. Reading out all channels every time two proton bunches cross would result in a data
rate of about 60 TB/s with an average event size of 1.5 MB. It is neither feasible to read nor to write data at this high rate, therefore, the rate is reduced drastically by the Level-1 hardware trigger and the online High Level Trigger farm (HLT). They discard most events and only select the physically most interesting ones, resulting in a data rate of about 1.5 GB/s. These events are now available for further processing and physics analyses.

**Physics objects reconstruction**

Physical objects in the event are reconstructed by the particle-flow method [14], using the combined information from all subdetectors. This includes the energy deposits in the calorimeters, as well as the tracks and vertices reconstructed from the tracker hits and muon system. By combining the spatial position of the different signals, tracks can be matched to corresponding energy deposits. Finally the causative particle can be identified together with its properties such as charge, energy and momentum. The particles are classified into five groups: electrons, muons, photons, neutral and charged hadrons.

**Jet reconstruction** – Possible constituents of a jet are clustered together by a jet algorithm. Within the CMS Collaboration the anti-\(k_t\) algorithm [15] is used most of the time. It is a sequential recombination algorithm that usually takes the particle-flow candidates as input. Particles are grouped together depending on their distances from each other and their transverse momenta. The result is a cone, ideally including all particles originating from the particle-level jet. Therefore, the four-momentum of the reconstructed jet consists of the sum of all constituents. If, however, a particle from the original parton is missed by the clustering algorithm, or a particle from another physics process is added to the cluster, the jet energy is affected and differs from the one of the original parton that initiated the jet. Chapter 7 describes the jet energy corrections accounting for these mismeasurements.

**Muon reconstruction** – The precise measurement of muons is relevant for many analyses, as well as the jet energy correction. Muons are identified by their hits in the muon system, as they are the only particles penetrating all previous detector layers. The information from the muon system is then matched to tracker hits for a precise measurement [4]. The charge of the muon is determined from the curvature of the trajectory.
The aforementioned event recording and reconstruction as well as the analysis and simulation of physics events require an enormous amount of computing resources. This is even more pressing since the start of Run II of the LHC, as the luminosity and therefore the data rates increased. This chapter introduces conventional computing paradigms and describes common HEP workflows. Furthermore, an established way of providing worldwide distributed computing resources is described.

3.1. Typical Workflows

Before describing the common HEP workflows, the two main computing paradigms are introduced as they are typically used to classify computing tasks and the corresponding hardware:

- **High Performance Computing (HPC)** – Focuses on the efficient execution of compute intensive, tightly-coupled tasks [16]. Common examples are weather forecasting, n-particle simulations in astrophysics and protein folding. The coupled tasks require a low latency connection between all compute nodes. This is what typical HPC centers are optimized for.

- **High Throughput Computing (HTC)** – Focuses on the efficient execution of a large number of loosely-coupled tasks [16]. Computing jobs can easily be partitioned and distributed on several independent clusters to reduce the overall runtime, as no inter-process communication is needed.

HEP workflows belong to the group of High Throughput Computing. The output of a single program running for months on a single computer is essentially the same as the output coming from multiple instances of the same program, each processing a piece
of the entire workflow. The jobs can be executed on physically distributed hosts and even at different points in time, as no inter-process communication is required. These characteristics enable the HEP community to run their workflows on a wide range of computing resources, even on HPC clusters.

HEP workflows themselves can again be classified into three groups:

- **Monte-Carlo event generation and detector simulation** – Simulation of a physics process e.g. proton-proton collisions at the LHC, interactions with detector material and detector response. The output of the full simulation procedure resembles the read out of the detector electronics. Basically no input is required, large amounts of data are written out, CPU intensive.

- **Event reconstruction** – Reconstruction of simulated or recorded events to obtain the physics objects and event parameters. For instance, applying the particle-flow reconstruction and anti-$k_t$ algorithms. These workflows require high input and output rates and are CPU intensive. In case simulated events are reconstructed this step can be combined with the previous one in order to reduce the amount of data to be transferred.

- **Physics analysis** – Analysis of physics events and objects, for example, the search for the Higgs boson or jet energy calibration. Typically the I/O rates are the bottleneck rather than the CPU utilization.

Because of the underlying nature of the described workflows, different requirements are imposed on the computing infrastructure. Therefore, the jobs are ideally processed on a computing farm providing the necessary high I/O rates and/or fast CPUs.

### 3.2. Computing Resources

Computing resources are typically accessed via a central batch system managing the available cluster machines as a pool of worker nodes and the computing jobs of users (see Figure 3.1). A user submits a list of computing jobs to the batch system’s queue. These jobs are then scheduled and distributed to free worker nodes, once they become available. The size of a batch system reaches from a small, local setup to a worldwide distributed system with hundreds of computing centers. A batch system is either interfaced directly via its own tools or by using a meta job-submission-tool such as GRID-CONTROL [17].

### The Worldwide LHC Computing Grid

The LHC community built a worldwide distributed, high throughput computing and storage infrastructure, known as the Worldwide LHC Computing Grid (WLCG) [19].
Initially it was designed in a layer-based structure centered around the CERN computing center, also referred to as Tier-0. This is where the online reconstruction of the raw data produced by the LHC experiments is performed. Both, the reconstructed and the raw data, are stored and distributed to thirteen Tier-1 centers around the world [20]. The centers serve as a backup and provide long-term storage. Additional reconstruction steps and Monte-Carlo event simulations are performed at these centers as well. The resulting datasets are transferred to multiple Tier-2 centers. They provide again storage and computing resources to the grid. While the access to the Tier-0 and Tier-1 resources is predominantly limited to selected people for reconstruction and production campaigns, the Tier-2 resources are open to grid users to process their physics analysis jobs.

### Local batch systems

Tier-3 resources, often referred to as local batch systems or computing clusters, mostly consist of university department clusters and are not formally engaged with the WLCG. They provide a vast amount of additional computing resources to end user physics analyses. Chapter 5 describes the dynamic extension of the local batch system of the Institute of Experimental Nuclear Physics (IEKP) at the Karlsruhe Institute of Technology (KIT).

### 3.3. Software Stack

The aforementioned HEP workflows require a well-defined software stack to ensure reproducibility of the processing on every single worker node contributing to the worldwide computing grid. This starts with a special operating system: Scientific Linux CERN (SLC), which is based on Red Hat Enterprise Linux [21]. It provides a common software baseline for all CERN experiments.

In addition, most of the required software for offline analysis is packed into the comprehensive CMS software framework (CMSSW) [22]. The framework centers
around the Event Data Model (EDM) concept, where the physics event is stored as an
C++ object container called Event. The definition of the data structure is provided, along
with a set of modules to analyze and filter Event inputs and store the processed output.
The package includes many classes and tools to perform various kinds of reconstruction
and analysis using recorded or simulated events from CMS. The deployment of new
versions and archiving of old versions for backwards compatibility is rather time
consuming and hardly manageable on a worldwide distributed computing resource. In
addition, the software comprises several gigabytes per release.

The CernVM project [23] provides a clever solution for this issue by decoupling
the operating system and the experiment software stack. The whole CMS software
is provided via the read-only file system CERNVM-FS. The required files for a user
job are downloaded on-demand via the HTTP protocol and are cached on the local
machine and on a site’s SQUID proxy. This method enables easy access to about 1 TByte
of up-to-date CMS experiment software [24], while the operating system itself only
requires about 1 GByte.

Besides the experiment software, tools providing interfaces to storage resources
such as SRM [25] or XROOTD [26] are made available via CERNVM-FS. XROOTD
can be used to stream experiment data from worldwide distributed storage elements
directly to a user’s analysis job without copying the files locally first. This follows the
Any Data, Anytime, Anywhere (AAA) paradigm of the CMS Collaboration [27] which
becomes more and more popular because of the increasing WAN speed between storage
and computing sites.

The described tools and technologies are used in Chapter 5 and 6 to enable high
energy physics computing in the cloud.
Chapter 4

Virtualized Computing Resources on Cloud Services

Today the buzz words cloud and cloud computing are common terms used not only in scientific and business domains but also in many areas of every-day personal life. This chapter gives an overview of the term cloud computing and its delivery models. Furthermore, the benefits of cloud computing and virtualization for the high energy physics community are discussed.

4.1. Cloud Computing and Virtualization

The cloud gives the illusion of infinite resources being available on-demand anywhere at any time. Resources refer, for example, to services such as an online storage or a search engine but also to virtual servers and entire computing infrastructures. A cloud provider maintains and manages these resources. Therefore, users do not have to care about the underlying server hardware, operating system, applications and not even about the physical location of the data center. Cloud computing features the pay-as-you-use billing method: Users pay only for the resources they actually use, instead of paying a fixed amount over time. This provides perfect conditions for any kind of business which can easily request a few cloud resources (e.g. for storage and computing) to develop a prototype and then scale the systems over orders of magnitude when required. Therefore, cloud computing minimizes the financial risks and reduces the time and effort needed to set up the desired systems [28].
4.1.1. Everything as a Service

The "as a Service" suffix is typical for any kind of cloud service made available over the internet. While there are terms like Business as a Service or Backup as a Service, all cloud services are related to one of the three levels of abstraction, the cloud delivery models, shown in Figure 4.1 [29].

**Infrastructure as a Service (IaaS)**

The IaaS model provides a user with computing infrastructure in the cloud. This may include computing nodes, storage elements and virtual networks. A user can easily combine and scale all components and build his own virtual data center. It is the model with the most control over the resources as the choice of operating system and software stack is left to the user. IaaS makes use of virtualization being further discussed in Section 4.1.2. Popular products are Amazon Elastic Compute Cloud (EC2) [30] for virtual machines and Simple Storage Service (S3) [31] for storage. OPENSTACK [32] and OPENNEBULA [33] are open source IaaS solutions which can be deployed on various kinds of hosts.

**Platform as a Service (PaaS)**

PaaS provides a programming or execution environment maintained by the provider. It is mainly used by developers for testing purposes or to deploy complex web applications requiring a MEAN stack (MONGODB, EXPRESS.JS, ANGULAR.JS, NODE.JS) or JAVA application servers (e.g. TOMCAT). PaaS is a trade-off between control and responsibility. Examples are Google’s App Engine [34] and Microsoft Azure [35].
Software as a Service (SaaS)

SaaS provides users with a single software or application while hiding the complexity of the underlying system. Access to such services is often provided via a web interface or a mobile app. Examples are cloud storage services such as DROPBOX [36], online applications such as Microsoft OFFICE ONLINE [37] and basically any Google service (Search, Mail, Maps, Calendar, ..).

4.1.2. Virtualization for the IaaS Model

Virtualization is the key component of the IaaS model: It allows to run several operating systems concurrently on the same physical hardware by encapsulating each instance in a so-called virtual machine. A special component on the host, a hypervisor, implements a layer between the physical hardware and virtual machines running the operating systems (guest system). The hypervisor manages the physical resources and provides each guest system with a virtual environment consisting of processors, memory, storage and, for example, a network interface. There are two kinds of hypervisors, Type-1 and Type-2. While Type-1 hypervisors run directly on the hosts hardware, Type-2 hypervisors such as KVM [38] or QEMU [39] run on a conventional operating system [29].

Virtual machines offer many benefits over running on a bare-metal system:

- **Availability** – In case a hardware failure occurs on a physical host, the virtual machine can easily be migrated to another host, even during runtime. Snapshots of virtual machines can be created to duplicate a machine or create a backup.

- **Separation of responsibilities** – Virtualization decouples the responsibilities of soft- and hardware maintenance. While a provider is in charge of the physical hosts and the hypervisor, a user can choose the desired operating system and software stack to meet the specific needs. An example for the HEP use case is given in Figure 4.2.

- **Sharing of resources** – The demand for computing resources can vary a lot, not only in science but also in many fields of business. For example, a company’s online store will serve more customers before Christmas time than usual. Virtualization allows to share physical hardware with different users and to easily scale the amount of resources.

An often discussed downside of virtualization is the reduced performance of the guest system compared to running applications natively. While this was true in the early 2000s, modern CPUs provide hardware accelerated virtualization functionalities reducing the performance penalty to below 10%. Furthermore, studies with CPU
intensive HEP applications have shown that the influence of the virtualization layer on the average execution time of computing jobs is negligible [40].

IaaS with **OPENSTACK**

**OPENSTACK** is an open source IaaS solution initiated by Rackspace and NASA in 2010 [41] and is backed by many major technology companies. It provides various components for virtualization, object storage, image services, etc. In addition, it works with several different hypervisors and emulators such as KVM or Xen. Its active community and open source nature make it the ideal IaaS solution for scientific projects. The cloud sites described in this thesis are all based on **OPENSTACK**.

4.2. Cloud Computing in High Energy Physics

To meet the ever growing demand for computing resources, traditional HEP-only clusters have been built and extended in the last years. However, the operation of private clusters for research groups requires planning and acquisition of hardware, finance for maintenance and capable administrators who manage the servers. To reduce costs and increase the operational efficiency, funding agencies nowadays encourage research groups to share computing resources provided by experts who operate computing clusters and data centers.

However, sharing resources requires the operators of the system to provide the required software environment for each user group. Fortunately, the evolution of cloud computing and the aforementioned benefits of virtualization open up new possibilities to outsource computing to a cloud provider while the user provides the desired HEP software environment (Figure 4.2). In addition, HEP jobs are suited perfectly to run inside a virtual machine as they do not require inter-process communication and are independent of each other.

Today, many commercial cloud sites are available and more and more scientific clusters use virtualization to ease the deployment of specialized software stacks. However, this requires the acceptance of the cluster operators as they are not always enthusiastic about the loss of control over the activities on their machines. The bwForCluster ENM at the University of Freiburg is discussed in Chapter 6 as a groundbreaking example of a dynamically virtualized shared cluster.
Virtualization allows separation of responsibilities: While the provider is in charge of the physical hardware and the virtualization technology (e.g. OPENSTACK), the user provides a virtual machine image containing the desired operating system and software stack. This enables provisioning of HEP infrastructure in the cloud.

**Figure 4.2.** Virtualization allows separation of responsibilities: While the provider is in charge of the physical hardware and the virtualization technology (e.g. OPENSTACK), the user provides a virtual machine image containing the desired operating system and software stack. This enables provisioning of HEP infrastructure in the cloud.
There are several arguments against a manual and static use of virtual machines. Manually booting several hundreds of virtual machines when required and shutting them down again is neither fun nor feasible. In addition, idling virtual machines block resources on a shared scientific cluster or might even increase operation costs when utilizing a commercial cloud site. Therefore, simple virtualization as described in the previous chapter is not sufficient for everyday use: The resource allocation needs to be fully dynamic.

In this chapter, the IEKP computing infrastructure is outlined, followed by a detailed description of the cloud meta-scheduler software ROCED and the integration of the institute’s new batch system. In addition, the build process of virtual machine images is explained. All in all, this comprises the methods and tools needed for dynamic provisioning of cloud resources as worker nodes for the institute’s computing pool.

5.1. Computing Infrastructure at IEKP

The computing infrastructure at the IEKP consists of desktop computers in a public network and a private network for the file servers and computing cluster. Users can access the private network via portal servers, which are also designated for development work, e.g. ekpcms6 for the CMS group.

The desktop computers provide the user with standard office applications such as web browser, text editor, programming IDE and a terminal. Even though most of the desktop computers are quite powerful nowadays and may serve as computing resources for smaller projects, it is not convenient to run a single instance of a program
for hours on a single machine, especially not when a batch system with access to a computing cluster is nearby.

The main batch system of the institute is based on Open Grid Scheduler [42]. It provides a static number of worker nodes based on SLC 5 and SLC 6 for the heterogeneous analysis software run by users. While the batch system software provides everything needed for a fixed-size cluster, it is not equipped with sufficient features to handle permanently appearing and disappearing worker nodes as it is the case when using cloud resources. Luckily, a new batch system, HTCONDOR, was already introduced for similar reasons, in particular to combine national resources for the German CMS community [43]. The new batch system has proven to be stable and flexible enough to replace the existing one. However, at the time of this writing, both systems are still coexisting, requiring users to choose between both resources. In the near future HTCONDOR will be used for all resources.

The local batch system: HTCONDOR

HTCONDOR [44] is a modular and very flexible batch system developed by the University of Wisconsin-Madison for HTC purposes. It is a distributed batch system using a client-server architecture which can handle tens of thousands of jobs using only a small managing machine for coordination of a compute pool. Two main features make it the batch system of choice for the integration of cloud resources. First, it easily handles the addition and removal of worker nodes to and from the pool. This is relevant as the cloud resources should only be available on-demand to handle peak loads. The second crucial feature is the TCP connection broker enabling access to resources in a private network or behind a firewall. This is especially relevant as public IPv4 addresses are rare and thus becoming more and more expensive and IPv6 is not yet wide-spread.

A HTCONDOR pool comprises a central manager and an arbitrary number of worker nodes. To make use of the pool, at least one submission node must be configured to allow jobs to be submitted to the pool, e.g. a portal server. A job consists of information about the executable and necessary input files, as well as requirements on the worker node. All machines constantly send a list of what they have to offer in form of so called machine ClassAds to the central manager. The manager then takes care of the matchmaking between all jobs and machine ClassAds. Besides simple job requirements for the amount of memory required, complex ranking and selection criteria using conditional expressions can be implemented to find the most suitable host for a job or to deny job execution when a given threshold, e.g. memory limit, is exceeded.

All hosts taking part in the HTCONDOR batch system run the condor_master daemon which spawns and monitors further daemons. A worker node additionally runs the condor_startd advertising the machine as an available resource to the central manager. Once a job is executed on the worker node, the condor_starter daemon is spawned and takes care of running and monitoring the job executable. At the time
Dynamic Integration of Cloud Resources into Local Computing Cluster

Listing 5.1: A simple JDL file to send 400 benchmark jobs to a new computing resource named bwForCluster.

| Executable = test_job.py |
| Universe = vanilla    |
| Log = test_job.log     |
| should_transfer.files = YES |
| Requirements=(TARGET.CLOUD_SITE="BWFORCLUSTER") |
| Queue 400              |

of this writing the HTCONDOR version deployed on the central manager and the virtualized worker nodes is 8.4.2.

Jobs can be submitted to HTCONDOR via the condor_submit <job.jdl> command. The job description file (jdl) contains all necessary information for executing the job. This includes the name of the executable, a list of files to be copied to the worker node and additional ClassAds to select the resource type where the job should be executed. Listing 5.1 shows a simple jdl configuration that was used to send benchmark jobs to a cloud site.

5.2. A Cloud Meta-Scheduler: ROCED

As by now, two of the three ingredients for successful dynamic provisioning of high energy physics infrastructure have been introduced: Virtualization and a batch system that can handle virtualized worker nodes. However, no central entity has been introduced yet, coordinating the cloud resources depending on the current demand for computing power.

For this purpose, the cloud meta-scheduler ROCED\(^1\) has been developed at the IEKP since 2010 [45–47]. As part of this thesis, the development of ROCED was resumed to cope with the rising demand for dynamic management of cloud resources. This includes adaption to new conditions in computing, e.g. handling of multi-core virtual machines, the integration of the new batch system and access to a new kind of cloud resource at a HPC center. This is later described in Section 5.2.2 and Chapter 6. As the software is being actively developed again and there is interest for such concepts by other groups, the code has recently been released on GitHub under the GNU General Public License [48].

\(^1\)Responsive On-demand Cloud Enabled Deployment
5.2.1. Idea and Concept

The idea behind the development of ROCED was to have a single tool that takes care of the on-demand provisioning of cloud resources as HEP worker nodes. The additional resources should be accessible by the user without major changes of the existing workflows. Ideally, the whole cloud infrastructure should be completely transparent to the user.

ROCED is written in Python and features a modular structure. By design there are three different kinds of Adapters implementing the main functionalities and being managed by the ROCED core. The core orchestrates all Adapters in a recurrent management cycle, e.g. every 30 seconds.

- **Requirement Adapter** – Supplies information about the demand for computing resources, for example, the number of waiting and running jobs in a batch system queue.
- **Site Adapter** – All communication between ROCED and a cloud site is handled by this Adapter. It takes care of booting, stopping and monitoring of virtual machines.
- **Integration Adapter** – Integrates booted machines into the batch system and removes them again before shutdown.

Adding new cloud sites or monitoring a different batch system is simply a matter of writing new Adapters which can then be integrated into ROCED. The central broker manages the information supplied by the Requirement Adapters and decides which machines at which cloud site should be started or stopped. For this decision it can take into account restrictions of the various cloud sites such as resource limits and pricing of cloud resources.

When ROCED is started, a configuration file is loaded containing at least one Adapter of each kind and its corresponding settings. These settings contain, for example, authentication information of a cloud site or parameters being used for querying a batch system. Multiple Site Adapters can be loaded in order to extend a single batch system with resources from multiple cloud sites.

Figure 5.1 illustrates the interaction between HTC/ONDOR, ROCED and several cloud sites. In addition, the log snippet in Appendix A.1 shows a running ROCED instance in action.

All Adapters share a common *machine registry* keeping track of all virtual machines managed by ROCED. It provides information about the machine state and any other property of interest, e.g. machine load, IP address and number of CPU cores. This allows to share information between different modules of ROCED. Machines are identified by their Universally Unique IDentifier (UUID) which is generated the first time a machine is added to the machine registry.
Dynamic Integration of Cloud Resources into Local Computing Cluster

It is essential to always keep track of machine state changes as this ensures that ROCED knows exactly what each machine is doing at each moment and may decide to intervene. The state may be updated by the Site or Integration Adapter, however, only linear changes in forward direction are allowed to ensure a structured lifetime of virtual machines. The possible machine states are the following:

- **booting**: machine was requested,
- **up**: machine booted successfully, but is not yet integrated into the batch system,
- **integrating**: currently integrating the machine into the batch system,
- **working**: machine is now a virtualized worker node, ready for job processing,
- **pending-disintegration**: machine won’t accept any new jobs, but will finish the running jobs (draining mode),
- **disintegrating**: machine finished all jobs and deregisters at the batch system,
- **disintegrated**: machine is still running, but no longer part of the computing pool,
- **down**: machine is shut down.

Figure 5.1: Dynamic integration of cloud resources into the local computing cluster. ROCED dynamically manages the number of virtual machines on different cloud sites, depending on the demand for computing resources. The integration of additional worker nodes is completely transparent to the end user. (Icons from [18].)
5.2.2. Monitoring the Demand for Computing Resources

The first step of integrating resources into the new HTCONDOR batch system was to develop and test the HTCONDOR Requirement and Integration Adapters. Their purpose is to deliver information about the demand for computing resources and integrate booted virtual machines into the batch system. HTCONDOR provides a powerful set of command-line interface tools to monitor and configure every detail of the central manager, worker nodes and computing jobs. Besides the command-line tools, there are also PYTHON bindings available allowing a direct access from any PYTHON code to HTCONDOR. However, these bindings are not well-documented and therefore the command-line tools are used in this thesis.

condor_q is the tool of choice to receive information about the jobs in the queue for the Requirement Adapter. The list of jobs is filtered to select only running and idle jobs, discarding failed jobs and jobs on hold. The output is a two-dimensional table where the first entry equals the job status and the second entry the number of requested CPU cores for this job. The respective command is executed via SSH on the ekpcm6 portal server:

```
condor_q -constraint 'JobStatus == 1 || JobStatus == 2' -format '%s,' JobStatus -format '%s\n' Requirements | grep <requirement> | awk -F ',' '{print $1","$2}'
```

where <requirement> is replaced by a configuration parameter, for instance, CLOUD_SITE=="BWFORCLUSTER" to select jobs suited for the bwForCluster cloud site. Using the requirement as a constraint in the condor_q call does not work as expected, therefore, the grep command is used. Finally, the total number of waiting and running jobs, multiplied with the number of requested cores, is passed on to the broker for decision making.

The Integration Adapter for HTCONDOR is not as complex as the ones for other batch systems. The aforementioned benefits of the new batch system make the integration of a virtual machine into the condor pool quite easy. No manual intervention is required. It is only necessary to start the condor_master daemon on bootup. The worker node will then connect to the central manager and advertise itself as a newly available resource. Because of this neat feature, the Integration Adapter only monitors the appearing and disappearing of machines in the HTCONDOR pool. In case a status change occurs, the Adapter takes care of changing the machine state from up to working or from working to pending-disintegration. The worker node pool is monitored using the following command line:

```
condor_status -constraint '<requirement>' -autoformat: Machine State Activity
```

where again <requirement> is replaced by the respective cloud site. The additional ClassAds State and Activity provide status information about each worker node slot
which is identified by the machine name. This information is used to calculate the load on the machine, i.e.

\[ I_{\text{machine}} = \frac{N_{\text{slots_claimed}}}{N_{\text{slots_available}}} \]  

(5.1)

which is then stored as a machine property and may serve as an input parameter when selecting the most suitable machines to terminate.

Several tests are implemented in both Adapters to ensure the correct handling of erroneous data that might be returned when querying the batch system. This mostly happens due to connection problems between the host running ROCED and the ekpcms6 server or when the HTCONDOR commands fail to query the central manager.

5.3. Building the Virtual Machine Image

The image of the virtual machine is a central component in the process of providing virtualized HEP infrastructure. The computing job of a user requires a well-defined software stack to be present, as well as tools and protocols to access WLCG resources. Furthermore, the tools and configuration to participate in the pool of worker nodes is required. The resulting image is essentially the same for multiple cloud sites, only differing with regard to some minor configuration parameters to adapt to the specific cloud site’s environment.

A common way to create such a virtual machine image is to boot up a virtual machine, install the required operating system, as well as the tools and configuration. Once the virtual machine meets the desired requirements, it is shut down and used as a base image for all future virtual machine instances. Doing all this manually requires at least one hour of supervision and is a rather monotonous work and error-prone. To speed up this process and to ensure perfect reproducibility of the image to provide always the same research environment, the virtual machine image is built in an automated way.

The OZ toolkit [49] was found to be the most reliable set of tools that is able to produce the required image and provide the flexibility for changes [40]. It boots a virtual machine, installs the required operating system and then opens a SSH connection to the guest system to install tools and the corresponding configuration. This is all done according to a template file, written in XML.

The following list shows the tools and modifications which are put into the image:

- **Operating system** – For a scientific computing environment at CERN, SLC 6.7 is the operating system of choice and is therefore used as a starting point for any virtualized worker node.
- **CERNVM-FS** – The CERN virtual machine file system provides access to a huge software repository, containing all the required libraries and tools to run physics analyses. In addition, tools and protocols for file access to the WLCG are provided. CERNVM-FS is installed and configured to use a SQUID proxy server close to the cloud site for caching reasons.

- **HTCONDOR** – Version 8.4.2 of the batch system client is installed, together with the required configuration files to connect to the central manager and act as a worker node.

- **De-contextualization** – Some modifications are required in order to be able to replicate the same image for all virtual machines and still come up with machines that have individual hostnames and IP addresses. In the case of OpenStack being the IaaS solution, CLOUD-INIT is installed, which is able to accept necessary configuration parameters passed to the virtual machine after booting.

- **Monitoring and management scripts** – Tools such as GANGLIA can be installed to monitor the internal state of the virtual machine. Besides that, management scripts for automated actions of the virtual machine are convenient. They are used for virtual machine images on the bwForCluster in Freiburg. Details are discussed in the following chapter.

- **SSH key** – For debugging it is helpful to install a SSH key or set a user password. The key can either be included into the image or injected into the virtual machine during boot time using CLOUD-INIT.

- **Final cleanup** – The last step of image creation consists of removing tools and files not being required for the desired use case. This includes, for example, clearing caches and removing man pages. Furthermore, the auto-update mechanism is disabled to prevent a distributed denial-of-service attack when 1000 virtual machines boot up at the same time and start downloading software updates.

The output of this process is a raw virtual machine disk image containing all required software and configuration. It is reasonable to compress the image down to about 1.6 GB and convert it to a more flexible file format before uploading it to the desired cloud site. In order to enable compression of the disk image, all free space of the guest file system needs to be filled with zeros by using the GUEST-FISH tool from LIBGUESTFS. Afterwards, the image is converted from the raw disk format to the qcow2 format using QEMU tools. This format supports the copy-on-write mode: In this mode, the basic image can be stored in a shared, read-only place, while all modifications during runtime are written to an overlay on a different location, e.g. the local disk of the virtual machine host. This feature is used for the OpenStack setup at the bwForCluster ENM in Freiburg being discussed in the next chapter.
CHAPTER 6

UTILIZATION OF CLOUD RESOURCES AT A REMOTE HPC CLUSTER

An integral part of this thesis is the dynamic integration of a new cloud resource located at a HPC cluster at the University of Freiburg. It was an outstanding opportunity to have a large scale cluster available for this purpose and to work with the HPC team of the data center of the University of Freiburg.

This chapter introduces the unique setup of the bwForCluster ENM and gives a detailed description of the transparent integration of cloud resources into the institute’s batch system, by using the tools and methods described in the previous chapter. The usability of these resources is studied by processing various HEP workflows on a similar prototype cluster. Finally, a long-term evaluation of the system’s stability is presented.

6.1. A Hybrid HPC Cluster: bwForCluster ENM

The bwForCluster ENM (Figure 6.1) will be one of the HPC clusters set up as part of the bwHPC-C5 project of the state of Baden-Württemberg [50]. The aim of this project is the coordinated operation of several HPC clusters for diverse scientific communities from the state’s educational institutions. In total there will be four HPC clusters. While the clusters in Ulm and Mannheim/Heidelberg are already in operation, the ones in Tübingen and Freiburg are still in procurement. All data centers are already connected with a 10 GBit/s link to the BelWü network [51], however, there are plans to upgrade the link speed to 100 GBit/s in the coming years. This allows fast data transfers between the clusters, the participating universities and the shared bwFileStorage, hosted at the KIT computing center.
Utilization of Cloud Resources at a Remote HPC Cluster

The cluster in Freiburg will provide high performance resources to three different scientific communities: Elementary Particle Physics, Neuroscience and Microsystems Engineering, hence the suffix ENM. Bringing these three communities together on a single cluster is not an easy undertaking. While neuroscience applications are typical HPC use cases requiring fast inter-process communication, particle physics analyses have special demands on the software stack as described in Section 3.3.

Even though the final cluster is still under construction and will be available in spring 2016, a testbed cluster called NEMO [52] is already available. It is running on old hardware from the bwGRiD cluster and provides 1248 CPU cores, which is only a fraction, compared to the 15000 cores on the final cluster. As this test cluster features the exact same configuration of the workload management system, it allowed intensive testing and evaluation of the complex process of dynamic provisioning of cloud resources. The methods and results developed as part of this thesis are essential for an instant usage of the new cluster once it becomes available.

Hybrid concept and technical details

To meet the demands from the three diverse user groups, the bwForCluster ENM features a setup where bare-metal computing and virtualization are coexisting on the same cluster. The hybrid approach meets the requirements in a unique way where no static partitioning of the cluster into a physical and virtualized segment is required. This is a completely new approach being one of the first of its kind.
The worker nodes run on CENTOS 7.2 and provide a comprehensive software stack for different fields of research. Users can connect to the login portal login.bwfor.uni-freiburg.de via SSH to prepare computing jobs and submit them to the cluster. The job scheduling is performed by the MOAB BASIC scheduler [53] providing the command-line tools msub, checkjob and mjobctl for job submission, monitoring and canceling.

Up to this point the setup is rather common for operating and providing HPC resources. The outstanding feature is the OPENSTACK based virtualization layer deployed on-top of the bare-metal cluster. This allows the user to run any kind of virtual machine on a worker node. However, OPENSTACK is not accessed directly by the user to start and stop virtual machines. Instead, the virtual machine scheduling is handled by the MOAB scheduler also taking care of the bare-metal jobs. This is implemented by a special user job containing a virtual machine request and is handled like any other user job. It enables a completely flexible coexistence of virtual machines and bare-metal jobs on the same hardware, respecting the user fair share and resource allocation on the cluster.

In order to make the MOAB scheduler cloud-aware, the Freiburg HPC team provides a PYTHON script which is submitted by the user to the Freiburg cluster and takes care of the instantiation of the virtual machine. Once the script is scheduled and executed on a worker node, it connects to the OPENSTACK manager and requests a virtual machine to be booted on the very same worker node that was allocated for the script itself. This is essential as the job scheduler must always know about the allocation of a worker node. Sending a user job to an occupied node might compromise the stability of the user job and the entire worker node as it might result in over-provisioning of the resources.

Besides the batch system access, the OPENSTACK dashboard is available to users. It can be used to upload the personalized virtual machine image and to view the status and log file (stdout) of the virtual machines. This feature comes in handy when trying to debug crashed or non-booting virtual machines. Furthermore, a special computing zone is provided, where static virtual machines can be booted. They run on separate hosts, not belonging to the pool managed by the MOAB scheduler and therefore do not have a limited run time (explanation in next paragraph). Such a service virtual machine could run a SQUID proxy server providing CERNVM-FS cache for the other virtual machines. In our case an existing proxy from the ATLAS computing group is used.

**Challenges**

Having virtual machines encapsulated in user jobs increases the complexity of managing these machines. As the underlying resources are shared with other communities, a virtual machine request might take several hours to get processed when the cluster is under heavy load. Additionally, the run time of virtual machines is limited to 4 days as this is the maximum wall time allowed for user jobs on the cluster, contrary to a regular
cloud site where static booking of virtual machines for several weeks is a common procedure.

### 6.2. Dynamically Managing Virtual Machines on a Hybrid HPC Cluster

Although the hybrid setup provides cloud-awareness to the MOAB job scheduler, it also adds an additional layer of complexity to the process of dynamic cloud resource management. Moreover, while the Freiburg HPC team acts as the provider and is in charge for the cluster and the virtualization layer, user groups interested in utilizing these resources are required to come up with a flexible approach of requesting the virtual machines and integrating them into their own workflows. The solution for this, provided as part of this thesis, requires the deployment and interaction of the following tools: ROCED and HTCONDOR on the HEP user side and OPENSTACK and MOAB on the provider side in Freiburg.

The following list describes the interaction of all components, illustrated in Figure 6.2. The technical details are discussed later on.

1. HEP user submits computing job to HTCONDOR e.g. via GRID-CONTROL.
2. ROCED continuously monitors the demand for computing resources.
3. If required, a batch job containing a virtual machine request is sent to the MOAB batch system in Freiburg.
4. The `StartVM` script is scheduled on the cluster and requests an OPENSTACK virtual machine on the allocated worker node.
5. After booting, the virtual machine integrates in the institute’s local batch system and acts as an additional worker node.
6. Jobs get scheduled on this virtualized HEP worker node.
7. Data can be read from and written to any HEP storage via common HEP protocols such as XROOTD for streaming and SRM for reading/writing.

This setup meets the two most important requirements when accessing cloud resources: **transparency** and **flexibility**. The cloud sites all provide the exact same environment to user jobs and are accessed via a commonly shared batch system. Accessing the resources is simply a matter of setting a single requirement parameter in the job configuration file (see Listing 5.1). Furthermore, the resources are allocated on-demand to run in a cost-efficient way.
The following paragraphs detail the technical implementation of virtual machine management on the cluster. All required features are integrated into a newly developed ROCED Site Adapter taking care of instantiation, monitoring and termination of virtual machines.

**Instantiation**

A virtual machine is requested by automatically connecting to the bwForCluster login portal via SSH and sending a batch job to the cluster containing the aforementioned StartVM script as a job executable:

```
msub -l walltime=4:00:00:00,mem=8gb,nodes=1:ppn=4 startVM_0.2.py
```

This command requests 4 CPU cores and 8 GByte of memory for 4 days. The number of cores and memory limit are used by the StartVM script to start a machine matching the respective boundaries of the batch job. The command returns the associated batch job ID which is stored in the ROCED machine registry. This allows identifying virtual machines when they connect to the HTCONDOR batch system using the batch job ID as a hostname and is furthermore required for the virtual machine termination and monitoring.

The StartVM script contains all logic required to communicate with OPENSTACK. It also provides some configuration parameters that are used for authentication and selecting the desired virtual machine image:
The USER_DATA variable allows passing information to the virtual machine at boot time via the CLOUD-INIT tool. In this case the requested job wall time is written to a file inside the virtual machine. The wall time is required for the automatic draining mode of virtual machines which is described in the following section.

Monitoring and managing

Monitoring the virtual machines includes detecting and recording status changes of virtual machines. During implementation and testing it turned out that simply monitoring the local pool of HTCONDOR worker nodes is not sufficient as there are many steps involved in requesting and booting a virtual machine which could fail at some point, this is described in more detail in Section 6.3.3.

Therefore, two tools of the MOAB batch system are used in addition, to check for running and completed jobs on the bwForCluster cluster:

```
showq -r -w user=<fr_username>
showq -c -w user=<fr_username>
```

A list of batch job IDs is returned which are linked to the respective virtual machine listed in the ROCED machine registry. While the first command is used to detect machines that left the MOAB batch queue and started running, the second one is used to detect virtual machines that were terminated unexpectedly without deregistering at the HTCONDOR batch system or failed to boot.

The aforementioned wall time limit of a user job on the bwForCluster requires managing the run time of a virtual machine in a way that running HEP computing jobs are not terminated together with a virtual machine when it reaches the wall time limit. To handle this situation, the so called drain mode of HTCONDOR is used: A job slot in drain mode does not accept any new jobs but finishes processing the active job. A script included in the virtual machine image monitors the current run time of the machine and sets its own job slots to drain mode once the machine’s run time reaches 85% of its maximum run time. This ensures that all jobs running less than 14.4 h are finished before the virtual machine is terminated forcefully. This limit can be adjusted to the requirements of the HEP job but is already sufficient for most cases. The draining of virtual machines is illustrated in Figure 6.4.
Termination

Two mechanisms of machine termination are used in this setup. One is implemented in ROCED, the other one directly inside the virtual machine. This allows for a combination of remote controlled shutdown and virtual machine self-management capabilities. Putting all logic in ROCED alone resulted in unsupervised virtual machines when the cloud manager suffered from connection issues. In this case the virtual machines need to react on their own to prevent unattended operation.

Depending on the state of a virtual machine, one of the two mechanisms applies:

- **booting state** – In case ROCED decides to terminate a virtual machine, the machines in booting state are selected to be shut down first. This is not uncommon since a shared cluster does not provide a fixed number of job slots which can result in multiple virtual machines being requested and held in the queue. These machines are terminated using the `mjobctl -c <job_id>` command of the MOAB batch system. This way the virtual machine request is canceled before the machine is ever booted.

- **working/draining state** – When a requested virtual machine is alive, ROCED will not perform the termination. The virtual machine contains a script regularly monitoring the state of the local HTCONDOR job slots. In case all slots, e.g. 4 of 4, have been idle for more than five minutes, the machine deregisters at HTCONDOR and shuts down. The `StartVM` script detects this, deletes the virtual machine and exits. The batch job in Freiburg is now finished and the resources are freed.

6.3. Tests and Evaluation

Orchestrating all tools to provide stable resources in a dynamic way is a complex and challenging task. There are many interfaces and connection points where something could go wrong and break the whole system. During the implementation of the required ROCED Adapters, many pitfalls were discovered and required special handling.

At the time of writing, the ROCED + bwForCluster setup has been in operation for five months. During this time many different HEP workflows were processed on this cloud site while the setup was monitored closely. The following use cases show the utilization of the Freiburg cloud and demonstrate several features of the setup.

6.3.1. Tests with Typical HEP Workflows

The different types of HEP workflows described in Section 3.1 were tested successfully on the described cloud resource.
Figure 6.3: Resource allocation over time of a skimming workflow using cloud virtualized worker nodes on the bwForCluster. ROCED takes care of requesting virtual machines. As soon as virtualized worker nodes become available, the user jobs are being processed resulting in less and less jobs in the HTCONDOR queue over time. Once the virtual machines are completely idle they start to automatically shut down and free up resources on the cluster.

Figure 6.3 shows the resource allocation over time of a data skimming workflow (medium CPU load, high I/O requirements) running on the bwForCluster. About 3300 computing jobs are submitted at once while no virtualized worker nodes are available. A limit of 680 requested CPU cores is set in ROCED, as the cluster currently has a hard limit of 640 cores per user. This way up to 20 virtual machine requests are queued ensuring fast scheduling in case a running virtual machine shuts down. However, not all 640 cores are available within 1–2 minutes after requesting, half of them take up to a day to get scheduled. This is due to a special way of limiting the resources per user on the cluster. More cores become available as time passes, resulting in a slow ramp-up of virtual machines during the first day of requesting them. This behavior might change once the final cluster is in operation.

Even though the main purpose of the resources in Freiburg are not data intensive jobs as they can overload the network connection, the skimming jobs worked flawlessly and were welcomed by the Freiburg HPC team as a load test and proof of concept. 30 TByte of data were streamed to Freiburg from multiple storage sites worldwide via XROOTD at a rate of 200–300 MByte/s, the output of 1 TByte was written to the GridKa data center in Karlsruhe.

Besides the skimming jobs, several diverse workflows of users were processed successfully on the cluster. This includes full event generation, detector simulation and reconstruction workflows. Figure 6.4 shows a part of a full CMS tt simulation at 8 TeV, illustrating the draining feature of the virtual machines. The computing jobs
Figure 6.4.: Excerpt of a full CMS t$t$t$t simulation workflow at 8 TeV center-of-mass energy. The virtual machines start draining before the underlying job reaches the wall time limit and the virtual machine is terminated. A newly implemented feature in ROCED accounts for the idle draining job slots being essentially free slots but are not usable due to the drain mode. Additional virtual machines are requested to provide the required amount of computing resources, which can be seen in the small increase of running virtual machines each time machines start draining.

used CERNVM-FS to access the CMSSW stack and query information required for the simulation procedure.

6.3.2. Long-Term Stability

The overall setup has been running stably for several months now and has been continuously improved as issues were encountered. Even though the cluster in Freiburg is still a testbed, it served well for many computing campaigns and set the way for the forthcoming final cluster.

Since the monitoring feature was implemented, 412 000 h of CPU time have been used on the cluster by the virtual machines. The complete usage overview is shown in Figure 6.5. The discontinuous usage of computing resources is typical for HEP users. While there are days with low workload, the demand for resources increases a lot in the weeks before important conferences or when new data from the experiments becomes available. This usage of resources perfectly illustrates why dynamic cloud resources are preferred over fixed-size private clusters: Sharing resources between multiple communities allows for a better overall utilization of resources and is more efficient. Moreover, ROCED can be used to combine the resources of several cloud sites to provide even more resources in times of high demand.
Utilization of Cloud Resources at a Remote HPC Cluster

Figure 6.5: Overall usage of cloud resources at the bwForCluster. The discontinuous behavior originates from the many different HEP workflows and submission scenarios. For example, testing the workflow with a few jobs and then submit the whole analysis later or increasing the size of a Monte-Carlo simulated sample by adding more computing jobs after a first batch has finished. Except for the one week vacation period at day 51 the other blanks represent times where the system was available but there was no work to be done.

6.3.3. Challenges and Issues

Several difficulties were encountered while implementing and monitoring the overall setup. The complex combination of HTCONDOR, ROCED, MOAB and OPENSTACK was not running smoothly from the beginning. However, the direct communication with the HPC team in Freiburg allowed for fast and uncomplicated solving of problems on both sides, Karlsruhe and Freiburg. The following list gives an overview of the challenges and problems which were encountered during this thesis and their respective solutions:

- **Network connection issues** – As communication via network connections is a key component of this distributed setup, ROCED needs to react to missing information about the queue size or number of running virtual machines in a reasonable manner. Missing information is usually caused by connection problems in the internal KIT network or to the bwForCluster. In case no information can be obtained, ROCED tries to wait for several management cycles. When the network outage lasts longer, the auto-shutdown mechanism prevents the virtual machines from blocking resources when they are inaccessible for the HTCONDOR batch system.

- **Worker node stability** – The utilized cluster being a testbed cluster running on old hardware, with the main focus not put on stability, it happened from time to time that virtual machines did not boot or died unexpectedly. ROCED is designed to be fault tolerant and handle these cases. The bwForCluster Site Adapter was also optimized in a way to monitor the status of the machines at different levels and...
tries to provide enough working virtual machines once some fail. Additionally, the capabilities of HTCONDOR come in handy, taking care of job resubmission once a worker node vanishes.

- **File system availability** – Instabilities of the NFS providing the base image to the virtual machines sometimes lead to file system corruptions inside the virtual machine and to new virtual machines failing to boot. The NFS shares in Freiburg were optimized, as well as the auto-shutdown mechanism of the virtual machines to work with a broken file system.

- **Detached virtual machines** – Several rare scenarios allowed to run virtual machines on the cluster being completely detached from the MOAB batch system. Some virtual machines were only partially associated, meaning there was an allocated worker node, but the virtual machine was booted on a different host. A mechanism on the bwForCluster now terminates any OPENSTACK instance not being associated with a certain batch job on the cluster. In addition, the user must strictly comply with the steps of requesting virtual machines defined by the provider.

### 6.4. Summary and Outlook

It is getting harder to justify the extension of private cluster systems in the basement of university departments. Sharing virtualized computing resources with other (scientific) communities allows easy buffering of peak loads and an efficient procurement of new resources. Additionally, separating the responsibilities for providing the computing resources and the specialized software stack allows the experts of both sides to work in an effective manner.

The bwForCluster ENM including its hybrid concept of combining bare-metal jobs and virtualization without hard partitioning of the hardware was introduced. The resources of this semi-public cloud were successfully utilized and embedded into the institute’s local computing cluster by improving ROCED and implementing the necessary Adapters. The complex management of virtual machines encapsulated in user jobs on the shared cluster is handled in an automated manner.

The new resources provide additional computing power for HEP analyses and large Monte-Carlo production campaigns. Long-term stability was proven by using the new cloud resources for over 5 months with several different kinds of HEP workflows. Moreover, the access to the resources is managed in a seamless and hassle-free way for the users, requiring no adaptations of their workflows.

Even though the implementation was done for the testbed cluster, the final cluster will soon be available and can be utilized by IEKP users right from the beginning. This is possible because of the extensive testing and implementation period allowing for a
good preparation. Besides that, the project benefited a lot from the good cooperation with the HPC provider team in Freiburg.

Once the final cluster becomes available, topics such as the user and group fair share need to be considered, since virtualization hides the actual HEP user behind a single account being responsible for the provisioning of the virtual machines. Additionally, backfilling mechanisms can be implemented so the flexible workflows may use idle resources on the cluster even though the current user fair share does not permit this. This requires that virtual machines can be terminated at any given time when the resources are needed by other users.

The integration of cloud resources located at the GridKa data center in Karlsruhe was also successfully studied during this thesis. Automating this integration process, as well as evaluating commercial cloud providers are future plans of the IEKP and are already work in progress. Furthermore, the use of Linux Containers, e.g. Docker, offers interesting opportunities by reducing the overhead of a virtual machine while providing the required software environment for user jobs on private resources.
The LHC being a proton collider, hadronic jets are the dominant signature in almost every collision event, i.e. jets are part of either signal or background processes. Therefore, the precise measurement of the jet energy scale is essential for almost all analyses performed at the CMS experiment.

Figure 7.1 shows a schematic overview of the evolution and detection of jets. Because of the complex nature of jets and their reconstruction (see Section 2.3) the jet energy measurement requires corrections for a wide range of different effects. While out-of-cone effects or initial- and final-state radiation can lead to a reduced measured energy, detector noise and additional simultaneous interactions within the same bunch-crossing (pileup) increase the measured jet energy.

This chapter introduces the CMS jet energy calibration procedure, followed by a detailed description of the new calibration software ExCALIBUR for absolute jet energy scale corrections. It was developed during this thesis for LHC Run II to benefit from a commonly shared codebase. The calibration procedure is illustrated by using the first accumulated data at a center-of-mass energy of 13 TeV. Additionally, a comparison between simulated events at 8 TeV and 13 TeV is presented which was performed during the preparations for LHC Run II using the newly developed calibration software.

7.1. CMS Jet Energy Calibration

To calibrate the jet energy scale at the CMS experiment, a multi-level correction approach has been established and is being used throughout the collaboration [55]. Three mandatory correction steps are consecutively applied on the jet momentum derived from particle reconstruction and jet clustering. In addition, several optional correction steps, e.g. flavor dependent corrections, can be applied. A schematic overview of the jet
**Figure 7.1.** Schematic overview of the jet evolution and measurement [54]. The partons originating from the hard interaction form color-neutral hadrons because of confinement. As these particles leave the beam pipe and enter the subdetectors, their energy, momentum and charge are measured. The trajectory of charged particles is measured in the tracker while the energy of most particles is measured in the electromagnetic and hadronic calorimeters. These energy deposits are later clustered by the particle-flow method [14] reconstructing particles using the combined information from all subdetectors.

**Figure 7.2.** Multi-level jet energy correction approach of the CMS Collaboration. For convenience the naming of the correction steps differs from the internal CMS jargon. The mapping is listed in Appendix B.3.

The energy correction procedure is shown in Figure 7.2. The corrections are derived from Monte-Carlo simulations and refined with data-driven methods. The former account for all known systematic deviations of the energy measurement while the corrections derived with data-driven methods account for effects not covered by simulation or theory predictions.
Step 1: Pileup offset corrections

At a collider with a high instantaneous luminosity such as the LHC, a recorded event does usually not only consist of a single hard interaction of two partons but contains additional softer proton-proton collisions. As these particles emerging from those so-called pileup collisions deposit additional energy in the calorimeters they may increase the measured raw momentum of a jet. The method used to subtract this excess is called the jet area method [56]. The corrections applied to measured and simulated events differ because of distinct pileup conditions in both samples.

Step 2: Simulation-based corrections

The bulk of the jet energy corrections is derived from Monte-Carlo simulations. The full simulation includes hadronization of the initial partons, simulation of the detector geometry and emulation of the read-out electronics. This way a precise prediction of the entire jet measurement procedure is possible. Another benefit of using simulation-based corrections is the accessibility of phase space areas that are usually less populated in data [56]. Overall, these corrections account for differences between parton level and reconstructed jets caused by out-of-cone effects, reconstruction inefficiencies for low energy particles and detector geometry effects, e.g. transition regions of detector modules.

Step 3: Data-driven residual corrections

Residual corrections are used to correct for the remaining differences between simulated and measured jet transverse momentum after the previous correction steps have been applied. There are two residual correction types, relative and absolute, both derived from data. The relative residual corrections are obtained by the analysis of di-jet events with one jet inside the barrel and one in the endcap or forward region. This method accounts for unknown inhomogeneities of the detector geometry not covered by detector simulation [55].

The absolute residual corrections represent the last step of the mandatory jet energy corrections. The correction factors are derived by comparing the reconstructed jet transverse momentum to that of a precisely measured reference object: a photon or Z boson. The absolute residual corrections at CMS are described in more detail in the following section.
Jet Energy Calibration with Z+Jet Events

7.1.1. Absolute Residual Corrections with Z(→μμ)+Jet Events

The absolute residual corrections make use of the precise measurement of a reference particle and link it to an object of interest (tag and probe method). The photon or Z boson are perfect candidates for this case as their energy is accurately measured in the ECAL (Z → ee, γ) or in the tracker and muon chambers (Z → μμ), respectively. This thesis focuses on events of a Z boson decaying into two muons as the reference object. Using muons benefits from the independence of the ECAL and its calibration in contrast to using photons or Z bosons decaying into an electron-positron pair, whose energy is measured by the ECAL.

Because of the unknown initial longitudinal momentum of the two colliding partons, it is not possible to derive the jet momentum from the reference object. However, following the law of energy and momentum conservation, the transverse momentum of both objects in Z+jet events must be equal as the initial transverse momentum of the collision is negligible. Figure 7.3 illustrates this so-called back-to-back topology of a jet and a Z boson decaying into two muons.

Jet response measurement

The jet response is defined to compare the measured transverse momentum of the jet with the one of the Z boson. In an ideal case with a correctly measured jet, the response would equal one. Within the CMS Collaboration, two complementary response methods are established [55].

Figure 7.3.: The ideal Z(→μμ)+jet topology: the Z boson is balanced by a single parton from the hard interaction. While the Z boson decays into two oppositely charged muons, the parton hadronizes and the resulting particles get clustered as a jet.
The $p_T$ balance method is defined as the transverse momentum ratio of the leading jet $p_T^{\text{jet1}}$ (jet with the highest $p_T$ value in the event) and the Z boson $p_T^{Z}$:

$$R_{\text{bal}} = \frac{p_T^{\text{jet1}}}{p_T^{Z}}.$$

This response is biased by final state radiation (FSR). A second jet emerging from the balancing jet reduces $p_T^{\text{jet1}}$ and thus lowers the measured jet response. A method to remove this bias is described later in this section.

The alternative Missing $E_T$ Projection Fraction method (MPF) takes all particles of the event into account by looking at the missing transverse energy $E_T^{\text{miss}}$. The missing transverse energy is defined as the negative sum of the transverse momentum of all particles in the event. This way it is less prone to FSR and out-of-cone effects. The method is based on the assumption that $Z(\rightarrow \mu \mu)$+jet events do not contain any intrinsic missing transverse energy, e.g. from neutrinos. Therefore, any variation in $E_T^{\text{miss}}$ is interpreted as a mismeasurement of the leading jet’s true transverse momentum. This interpretation, however, requires a fully aligned and calibrated detector as this influences the $E_T^{\text{miss}}$ measurement as well and thus might affect the jet response. The MPF response is defined as

$$R_{\text{MPF}} = 1 + \frac{E_T^{\text{miss}} \cdot p_T^{Z}}{|p_T^{\text{jet1}}|^2}.$$

The ideal $Z$+jet topology consisting of a single jet balancing the Z boson is extremely rare because of initial and final state radiation, pileup and underlying event activity. This can be accounted for by introducing a new variable quantifying the amount of additional activity in an event. This quantity $\alpha$ is defined as the ratio of the transverse momenta $p_T^{\text{jet2}}$ of the second leading jet and the Z boson

$$\alpha = \frac{p_T^{\text{jet2}}}{p_T^{Z}}.$$

$\alpha$ can either be used as a selection criteria to limit the deviation from the ideal topology or for extrapolation towards negligible second jet activity ($\alpha \rightarrow 0$) in order to retrieve the absolute residual jet energy correction values. This procedure is further described in Section 7.2.7.
7.2. Calibrating the Absolute Jet Energy Scale with $Z(\rightarrow \mu\mu)$+Jet Events

7.2.1. The new Calibration Software

A fast turnaround time of jet energy corrections is mandatory to provide the CMS Collaboration with the correction factors for the latest accumulated data in a fast and flexible manner. To meet these demands, a well structured and modular software is required.

During the LHC Run I calibration software was developed and used successfully by members of the IEKP. The long-term shutdown period between Run I and Run II was used to develop a general event-based data-processing framework at the IEKP to provide a common base for various kinds of physics analysis. In order to prepare for Run II, a major goal of this thesis was the development and implementation of a new $Z(\rightarrow \mu\mu)$+jet energy calibration software, based on the new general analysis framework. The new resulting calibration software named EXCALIBUR was successfully developed and tested. Its implementation and underlying physics motivation are described in detail in this section.

During the set-up of the EXCALIBUR software and the implementation of the individual modules, the output of the analysis was continuously validated. To do this, the properties of the event and physics objects were compared to the output from the former calibration tool used for LHC Run I. Reproducing the calibration results of Run I data is essential to ensure that everything was set up correctly and is ready for the LHC Run II data-taking. For this purpose, an event matching tool was created, comparing up to three ROOT ntuples and matches them based on lumi-section, run and event number. The tool allows for a detailed synchronization of all properties stored in the input files.

Together with this tool, an intensive synchronization of the former and new calibration tools was performed using 8 TeV recorded and simulated events. The results of the former software were reproduced successfully, up to the precision of the data types used for storing the respective quantities. This in-depth synchronization allowed to identify several minor flaws in the former calibration software either not being present in the shared modules of the underlying new framework or fixed during the implementation of new modules as part of this thesis. This synchronization clearly showed the advantages of a shared base framework and of in-depth code reviews of existing code.

An event-based data-processing framework

A great advantage of a shared framework are the synergy effects that come along with it. Different analysis steps can be shared among analysis groups and people can
benefit from the effort of others who have already contributed the necessary building blocks in the form of modules. This allows to quickly set up an analysis without having to reinvent the wheel every time. Furthermore, it enables joint adaption and synchronization when central parts such as the physics object identification criteria change. Moreover, there is a big gain in code quality due to the many users using the codebase and reviewing the commits.

To benefit from such a common tool, a modern and flexible event-based data-processing framework called ARTUS has been developed by members of the IEKP [57]. The concept of the framework was taken from the former jet calibration software of the group [54]. The framework is written in a highly modular manner using C++11 and PYTHON. It comes with a set of modules for object reconstruction, filtering and storing of output, which can be enabled and configured via options set in a configuration file.

An important feature of the framework is the possibility to run the same analysis with different settings on the same event in so-called pipelines. This has the advantage that the event is only loaded and processed once for several different configuration settings. This allows fast processing when comparing the results for different analysis parameters e.g. physics object selection depending on kinematic quantities.

Producers are modules providing all the necessary tools for physics object validation and correction and provide methods to calculate derived quantities based on the output of preceding producers or the event content. A set of filters is available to veto events and physics objects that do not pass certain quality criteria such as trigger decisions or selection criteria on physics objects parameters. In the last step of processing, the relevant quantities generated by the producers or read directly from the event are written to the output file, typically a ROOT tree, by a consumer. This is done separately for each pipeline containing a specific analysis configuration. The aforementioned workflow is illustrated in Figure 7.4, for further documentation see [58].

The reduced sample of selected events provided by ARTUS can be further processed to obtain histograms in publication quality. For this purpose, the ARTUS framework includes the powerful plotting tool HARRYPLOTTER to generate histograms and fits from ROOT input files using either MATPLOTLIB or ROOT for plotting. Within seconds, additional selection criteria can be imposed and dozens of plots are generated.

The input files for ARTUS, the skims, are produced by the skimming tool KAPPA, which is shared among different groups as well. It is developed at the IEKP and acts as a stable interface to the permanently changing versions of CMSSW, allowing a easy change to the latest CMSSW versions and different data input formats (miniAOD, AOD, RECO) [59]. The output files contain all relevant physics objects and event meta-data required for a specific analysis.
The modular structure of an ARTUS analysis (based on [54]). All available objects and quantities of the physical event are provided to the producers. Additionally, the producers store their output in the so-called product which can be accessed again by all other (global) producers.

**The Z(\(\rightarrow \mu \mu\)+jet calibration software: EXCALIBUR**

During this thesis, the existing jet energy calibration tool used for LHC Run I was ported to the new ARTUS framework to benefit from the shared codebase. To achieve this, the existing producers and filters of ARTUS were evaluated regarding the possibility of using them for jet energy calibration purposes. While some of the modules could be used out of the box, only requiring configuration settings, others needed to be implemented into the new calibration framework [60] as they where analysis specific. An overview of all the required modules for the jet energy calibration is given in Figure 7.5.

A basic configuration file loads general Z+jet modules providing a muon and jet collection, Z boson reconstruction and jet energy corrections from previous levels. A flexible set of configurations is then generated by PYTHON scripts using the base file. This implies, for instance, loading of Monte-Carlo specific modules such as several generator object matching producers. The result is an easy generation of new configuration sets for processing of different input types such as data or Monte-Carlo samples with center-of-mass energies of 8 or 13 TeV.

Furthermore, multiple pipelines are generated gradually enabling filters imposing thresholds on different kinematic quantities of the physics objects. This facilitates studying the impact of selection criteria on all the quantities provided by the framework. The usage and implementation of the modules is described in the following sections together with their physical motivation. The order of the modules is to some extent based on the best runtime performance rather than their physics context, despite the latter might be more intuitive at first glance. Running fast event and object filters first reduces the overall run time as discarded events leave the pipeline at an early stage and are not processed by the subsequent slower modules.
The calibration procedure is illustrated by using the data recorded in 2015 with a center-of-mass energy of 13 TeV. Only the data-taking periods C and D with a bunch spacing of 25 ns are used, corresponding to an integrated luminosity of $L = 2.11 \text{ fb}^{-1}$. Detailed information about the data as well as the simulated events is listed in Appendix B.1.

### 7.2.2. Event and Physics Objects Selection

#### Event selection

Not all events recorded by the CMS detector are eligible. To select events recorded when the detector system was in nominal operating mode, the CMS Physics Validation group regularly releases a JSON file containing valid run numbers and lumi-sections. The run/lumi-section combination is then used to select the valid events from the input.
files. In ARTUS this is carried out by the JsonFilter usually being placed as the first module since it has a reasonably short runtime and reduces the amount of events to be processed by the following modules significantly. Around one-fifth of the events of the year 2015 dataset were rejected by this filter.

The High Level Trigger farm (HLT) reduces the number of events to be stored and provides trigger decisions for analysis purposes [61]. This can, for example, be a double muon trigger used for the jet energy calibration. It enforces a threshold of 17 GeV and 8 GeV for the transverse momentum of the leading and second leading muon respectively. The HltProducer and HltFilter are run last in the global module chain as the trigger selection uses regular expressions for the name matching which is quite CPU intensive.

**Physics objects selection**

In addition to the event selection, the objects constituting the event need to be validated. Most importantly, the jets and muons are checked for their authenticity, as they can be misidentified as other particles or erroneous reconstruction. For the muons this is done by selecting only muons agreeing with the recommendations provided by the Muon Physics Object Group, the so-called muon identification (MuonID) [62]. While the loose and medium MuonID require only few weak criteria to select valid muons, the tight MuonID is used for this analysis to get the purest sample of muons. It requires, for example, a minimum of five hits in the tracker to guarantee a good $p_T$ measurement. All criteria are listed in Appendix B.3.

Furthermore, the muons are required to be isolated from other particles in order to reject non-prompt muons from jets. These isolation methods, as well as the different MuonID levels which may vary for different data-taking periods, are available by the ValidMuonsProducer and shared among ARTUS users. This allows to analyze 8 and 13 TeV inputs with the corresponding physics objects IDs by simply changing a configuration setting.

The selection of valid jets works in a similar manner. The jet identification (JetID) recommendations are provided by the JetMET Algorithms and Reconstruction Group and currently come in two different flavors [63]. The ValidJetsProducer selects jets according to the selected recommendation. For the jet energy calibration the loose JetID is used to create a preselection of jets. The JetID prunes the collection of jets in the event, removing any which likely do not represent an actual parton shower.

Only after the jet and muon identification, the remaining event topology is validated again by the MinNMuonsCutFilter, MaxNMuonsCutFilter and ValidJets Filter. These filters discard events containing less than two or more than three valid muons and require the event to contain at least one valid jet. As events with more than three muons are very rare, they are discarded to enable a more simple reconstruction of the $Z$ boson.
The JSON, HLT and ID modules described so far are global and affect all configuration pipelines. Further kinematic selection criteria are applied in a later step on a per-pipeline basis, after additional global modules are executed which are required by all pipelines (see Section 7.2.6).

**Z boson reconstruction**

The ZProducer reconstructs a Z boson candidate from any two oppositely charged muons. Besides the charge requirement, the invariant mass of the muon pair must be close to the value of the Z boson mass provided by the Particle Data Group (PDG) [64]

\[
m_{\mu\mu} - m_{Z}^{\text{PDG}} < 20 \text{ GeV}
\]  

(7.4)

If no valid Z boson can be reconstructed with this procedure, the event is discarded by the ZFilter. Figure 7.6 shows the mass distribution of the reconstructed Z bosons.

**7.2.3. Applying Previous Jet Energy Corrections Steps**

To determine the absolute residual correction factors it is necessary to first apply the corrections from the previous steps 1 and 2 as described in Section 7.1. For closure tests it is also possible to activate the step 3 corrections for data inputs, consisting of all absolute residual corrections including the \(Z(\rightarrow \mu\mu)+\text{jet}\) corrections from this analysis. Furthermore, corrections for the \(E_T^{\text{miss}}\), the so-called Random Cone (RC) corrections, can be applied correcting for the average energy offset due to pileup in data samples [56].
The different correction steps are provided as text files by the Jet Energy Resolution and Corrections Group (JERC), separately for data and Monte-Carlo samples. The files do not simply consist of a single correction factor but include a non-trivial formula to take into account the angular position $\eta$ of the jet and the jet’s transverse momentum $p_T$. This way the jets are corrected individually depending on the pileup density $\rho$ of the event, the jet area $A_j$ and the jet’s momentum $p_T$ and pseudorapidity $\eta$.

In the daily business of deriving the absolute jet energy corrections it is often necessary to study and cross-check the impact of the other correction levels. To simplify this process the ZJetCorrectionsProducer copies all jets to a separate jet collection after applying the next correction step. This is easily accomplished by the pipeline mechanism of the framework. A mapping between the names of the correction files and the naming within the EXCALIBUR framework is given in Appendix B.3.

7.2.4. Missing Transverse Energy Correction

The mismeasurement of the jet’s constituents does not only affect the energy measurement of the jet, but also influences the $E_T^{\text{miss}}$ of the whole event. To account for this, the Type-I $E_T^{\text{miss}}$ correction propagates the changes of the jet energy corrections to the missing transverse energy.

This is performed by the TypeIMETProducer iterating over all jets in the event with a $p_T$ above 15 GeV and summing up the differences between the step 2 (for recorded and simulated events) or step 3 (recorded events only) and the RC corrected jets. This sum is then subtracted from the raw $E_T^{\text{miss}}$ to retrieve the corrected $E_T^{\text{miss}}$ of the event. The RC corrections are used instead of the step 1 offset corrections to account for varying pileup situations inside and outside the jet cone.

Prior to this step the corrected jets are not sorted by $p_T$ as the order needs to be preserved for the $E_T^{\text{miss}}$ corrections. Afterwards the JetSorterProducer takes care of sorting the jets in the individual collections as the order might have changed because of the varying jet correction factors.

7.2.5. Event Weight Calculation

Some of the parameters used for creation of a Monte-Carlo sample do not reflect the actual experimental situation at the time of data-taking. To account for this, parameters such as the pileup distribution need to be reweighted to enable a comparison between simulated and measured events. Further normalization factors account for differences in the cross section and luminosity. Those event and sample weights are calculated by different producers and then merged into a single weight by the EventWeightProducer. Additional weights can also easily be applied on plotting level using HARRYPLOTTER.
As an example for a weight producer the PUWeightProducer is described here. The pileup distribution per event varies during data-taking as it depends on various beam parameters changing over time. Thus it cannot be included correctly in the simulation, usually being performed before or during data-taking. Reweighting the simulated events by their mean number of pileup vertices

\[ \langle n_{\text{PU}} \rangle = L \cdot \sigma_{\text{inelastic}} \]  

provides pileup conditions close to the ones actually observed. The information required for reweighting is provided in the form of a ROOT histogram. This histogram in turn is created by a PYTHON script using the bunch-by-bunch luminosity \( L \) in data, provided by the CMS Collaboration, and the total inelastic cross section \( \sigma_{\text{inelastic}} \approx 69.0 \text{ mb} \) as input parameters. For comparison reasons, the mean number of expected pileup vertices is also provided in the data sample by the NPUPProducer, using the same input parameters.

### 7.2.6. Specific Selections for the Z+Jet Topology

In addition to the basic event selection, the muon and the jet validation, analysis specific selection criteria are applied to further improve the purity of the selected events for calibration purposes. The selection criteria are applied on a per-pipeline basis to allow studying of all output quantities at different selection levels. In case a quantity of a physics object does not meet the requirements, the whole event is discarded. For illustration purposes, all following plots show the distribution of the quantity in question using simulated events at 13 TeV. No further selection criteria are applied, besides the preselection on skim level. Therefore, some events might be discarded by more than one selection criteria.

The transverse momentum of the muons is required to be slightly above the trigger threshold of 17 GeV to avoid trigger turn-on effects. Additionally, only muons in the central part of the detector are selected as the muon system does not reach beyond \( |\eta| > 2.4 \). Hence the following selections are applied by the MuonPtCutFilter and MuonEtaCutFilter

\[ p_T^\mu > 20 \text{ GeV} , \]  
\[ |\eta^\mu| < 2.3 . \]  

These selections reject only a few events, as restrictions of the double muon trigger are already active at this time and the reach of the muon system is limited. The effect of these selections on the respective quantities is shown in Figure 7.7.

The leading jet needs to be reconstructed in the homogeneous central part of the detector and tracker. Additionally, a limit on the transverse momentum is defined to reduce the influence of detector noise and pileup. Beyond that, a lower threshold
Jet Energy Calibration with Z+Jet Events

Figure 7.7: Distribution of transverse momentum (left) and pseudorapidity (right) of the leading muon. The greenish area indicates the range being excluded. Only a few events are rejected because of the active preselection of the double muon trigger and the limited reach of the muon system.

on the transverse momentum is applied to all other jets in the event by $\kappa_{\text{APPA}}$ on skimming level [54]. The other selections are applied by the LeadingJetPtCutFilter and LeadingJetEtaCutFilter, their impact is shown in Figure 7.8:

\begin{align*}
|\eta_{\text{jet1}}| & < 1.3 , \\
\pT_{\text{jet1}} & > 12 \text{ GeV} , \\
\pT_{\text{jet1}} & > 5 \text{ GeV} \quad \text{(rejects jets, not the whole event)} .
\end{align*} \tag{7.8, 7.9, 7.10}

While the limit on the transverse momentum only affects a few events, many are discarded because the jet was not reconstructed inside the central and homogeneous part of the hadronic calorimeter.

To ensure the ideal $Z(\rightarrow \mu\mu) + \text{jet}$ event topology described in Section 7.1.1, the leading jet and the Z boson are required to point into opposite directions in the $\phi$-plane of the detector. In addition, selection criteria on the $\alpha$ distribution and the transverse momentum of the Z boson are imposed to lower second jet activity and the influence of low energetic particles, respectively. This is ensured by the BackToBackCutFilter, AlphaCutFilter and ZPtCutFilter using the following selection criteria, whose impact is shown in Figure 7.9:

\begin{align*}
|\Delta \phi(Z, \text{jet}_1)| & < 0.34 , \\
\alpha = \frac{\pT_{\text{jet2}}}{\pT_Z} & < 0.3 , \\
\pT_Z & < 30 \text{ GeV} .
\end{align*} \tag{7.11, 7.12, 7.13}
Figure 7.8: Distribution of transverse momentum (left) and pseudorapidity (right) of the leading jet. The greenish area indicates the range being excluded. The two spikes in the pseudorapidity distribution originate from detector inhomogeneities in transition regions of subdetectors.

Even tough the $\alpha$ and $\Delta \phi$ selections both reject most events, they often affect the same events as both selection criteria are partially correlated. A method to reduce the amount of rejected events by the $\alpha$ threshold, using a new pileup identification method, is described in [65].

A histogram showing the absolute and relative impact of all filters is generated by the CutFlowHistogramConsumer. Its result for the whole calibration analysis is shown in Figure 7.10.
Jet Energy Calibration with Z+Jet Events

Figure 7.9: Distribution of $\Delta \phi$ of the leading jet and Z boson (top left), $\alpha$ (top right) and transverse momentum of the Z boson (bottom). The greenish area indicates the range being excluded.

Figure 7.10: Absolute (left) and relative (right) selection efficiency of each filter. The huge impact of the MinMuonsCutFilter on recorded events compared to simulated events originates from the fact that the simulated events are based on a Drell-Yan sample, while in recorded events the two muons could originate also from other processes.
7.2.7. Jet Response Determination

After cleaning the input samples and validation and correction of the physics objects, the jet responses defined in Section 7.1.1 are retrieved to perform the actual $Z(\rightarrow \mu\mu)$+jet energy calibration. While the $p_T$ balance response is calculated on the fly during plotting, the MPF response is calculated in EXCALIBUR and stored by the ZJetTreeConsumer among all other event and physics object’s properties of interest. The responses are shown in Figure 7.11 and 7.12 with the correction steps 1 and 2 applied, to study the absolute residual corrections.

The agreement between simulated and recorded events is better for the $p_T$ balance responses than it is for the MPF responses. The differences might be the result of a not perfectly aligned and calibrated detector as the MPF method is affected by any kind of mismeasurement of the $E_T^{\text{miss}}$. This results in higher absolute residual correction factors when the extrapolation is performed in the next step. In both cases the responses are stable with regard to the number of reconstructed vertices indicating a good removal of pileup vertices.

While it is important to study the response dependence on $n_{PV}$ and $p_T^Z$, these results are not used for the calibration as they still contain events with non-negligible second jet activity from final-state radiation. To remove this bias, an extrapolation is performed, illustrated in Figure 7.13.

![Figure 7.11](image.png)

**Figure 7.11:** The $p_T$ balance jet response over the number of reconstructed vertices $n_{PV}$ (left) and the transverse momentum of the Z boson $p_T^Z$ (right). The responses from recorded and simulated events mostly agree within the statistical uncertainties. The flat distribution over $n_{PV}$ indicates good removal of pileup, while the lower response for low values of $p_T^Z$ originates from poor reconstruction and additional out-of-cone effects.
The jet response is extrapolated from non-negligible second jet activity to the ideal Z+jet event topology ($\alpha = 0$). This only affects the $p_T$ balance response, while the MPF response is predominantly independent by design. The absolute residual correction factors are derived from the ratio of recorded and simulates events at $\alpha = 0$. In the limit of the perfect event topology both methods agree well within the estimated uncertainties. The obtained response ratios are

$$R_{\text{data/simul.}}^{\text{bal}} = 0.998 \pm 0.005 ,$$  \hspace{1cm} (7.14)

$$R_{\text{data/simul.}}^{\text{MPF}} = 0.990 \pm 0.005 .$$  \hspace{1cm} (7.15)

Both response ratios are already close to one. This originates from the fact that the bulk of the jet energy corrections is already applied in step 2 and only minor residual corrections are required.

For combination of the absolute and relative residual corrections from other groups, a global fit is performed. The resulting common set of residual corrections is then published within the CMS Collaboration to achieve fully corrected jets for other analyses. The complete procedure of combining all correction factors is described in [56], along with the global fit method used.
Jet Energy Calibration with Z+Jet Events

Figure 7.13.: Extrapolation of the jet response to the ideal event topology to retrieve the absolute residual corrections (left) and with all correction steps applied for closure tests (right). The relevant correction factors are derived from the ratio of recorded and simulates events at $\alpha = 0$, where both responses are already close to one. The closure test on the right, including the absolute and relative residual corrections, shows a slight overcorrection for both responses in this case.

7.3. Comparison of Simulated Events at 8 and 13 TeV

An import part of the jet energy calibration group preparations for LHC Run II was the first analysis of Monte-Carlo samples at 13 TeV center-of-mass energy. Studying these samples as a part of this thesis allowed to perform two task at once:

1. test the new Z+jet framework with 13 TeV input,
2. study the effects of the increased center-of-mass energy of 13 TeV on the absolute residual corrections.

Besides the increased energy, the spacing between two proton bunches was reduced from 50 ns to 25 ns to increase the luminosity of the LHC. To account for the change in the bunch spacing, the 13 TeV sample is reweighted based on the number of pileup vertices of the other sample. This is similar to the pileup reweighting described in Section 7.2.5. The corresponding distributions before and after reweighting are shown in Figure 7.14. In addition, all 13 TeV distributions are normalized to the 8 TeV distributions to account for the slightly different sizes of the Monte-Carlo samples. Technical details about the samples are listed in Appendix B.1.

However, not only the center-of-mass energy and proton bunch spacing changed, also many parts of the detector hardware were replaced and upgraded and the reconstruction software was improved. While some of the changes such as enhanced
reconstruction efficiencies or upgraded detector cells are known to affect the measurement of physics objects properties, there might also be unknown effects, e.g. due to detector misalignment. This may result in additional differences between the two compared samples.

7.3.1. Object Properties

Figure 7.15 shows the transverse momentum and pseudorapidity distributions of the leading muon. All selection criteria for muons (see Equation 7.6 and 7.7) are enabled. The transverse momentum is shifted towards higher energies in the 13 TeV sample. This is expected because of the increased center-of-mass energy of the LHC. This also affects the pseudorapidity distributions, which is slightly broader as more muons outside the central detector have enough transverse momentum to survive the trigger selection. The same changes also apply to the trailing muon, the respective distributions are listed in Appendix B.4.

The transverse momentum and mass of the reconstructed Z boson are shown in Figure 7.16, all selection criteria listed in 7.13 are applied. The increased center-of-mass energy again shifts the transverse momentum distribution towards higher energies. Even though the energy resolution of the muon system decreases with increasing energy, the mass distribution is slightly sharper for the 13 TeV sample. This is potentially due to the improved muon reconstruction.
Figure 7.15.: Comparison of leading muon’s transverse momentum (left) and pseudorapidity (right) at 8 and 13 TeV. As expected by the increased center-of-mass energy, the transverse momentum is shifted towards higher energies while the pseudorapidity distribution is broadened. Both observations also apply to the trailing muon, see Appendix B.4.

Figure 7.16.: $Z$ boson transverse momentum (left) and mass (right) distributions. The increased center-of-mass energy shifts the $p_T^Z$ distribution towards higher energies and the mass distribution is slightly sharper for the 13 TeV sample.

For comparison of the leading and second leading jet properties, the corresponding selection criteria listed in Equation (7.8–7.10) are enforced. The jets are fully corrected using step 1 and 2, step 3 is omitted as it is not relevant for simulated events. The leading jet properties are shown in Figure 7.17, additional quantities are listed in Appendix B.4. While the transverse momentum distribution is again shifted because
Jet Energy Calibration with Z+Jet Events

Figure 7.17: Comparison of leading jet’s transverse momentum (left) and pseudorapidity (right) at 8 and 13 TeV. The transverse momentum distribution is shifted because of the additional center-of-mass energy, the pseudorapidity does not change within the statistical uncertainties.

of the additional center-of-mass energy available, the pseudorapidity does not change within the statistical uncertainties.

The relative increase of transverse momentum of the second leading jet is larger than the one of the leading jet at the higher center-of-mass energy. This results in the $\alpha$ distribution being shifted towards higher values, as shown Figure 7.18. During LHC Run I all events with $\alpha > 0.2$ were discarded. This threshold is loosened up to 0.3 for the Run II jet energy calibration in order to discard less events, but kept at 0.2 here for the comparison.

7.3.2. Jet Response

The jet responses are shown for comparison only, the extrapolation to zero second jet activity is not performed, as only simulated events are being studied here. The response dependence on the transverse momentum of the Z boson is shown in Figure 7.19. The increased center-of-mass energy lowers the $p_T$ balance response, potentially because of additional out-of-cone effects and increased radiation. This was already indicated by the increased share of the second leading jet transverse momentum. The MPF response, however, is not sensitive to these effects as it takes the whole event into account. The removal of pileup vertices remains at a similar good level as it was during Run I, the distributions are shown in Appendix B.4.

Even though the complex detector and the corresponding software underwent many changes during the upgrade period between Run I and Run II, the detector seems well
Figure 7.18.: The increased transverse momentum of the second leading jet (left) results in a shift of the $\alpha$ distribution (right) towards higher values.

Figure 7.19.: $p_T$ balance (left) and MPF (right) jet response over the transverse momentum of the $Z$ boson $p_T^Z$.

calibrated an understood from the Z+jet calibration point of view. The observed effects can be explained by the increased center-of-mass energy. In addition, the EXCALIBUR Z+jet calibration software proved to successfully process 13 TeV simulated events and is ready for the LHC Run II.
7.4. Summary

The need for a jet energy calibration was motivated and the CMS multi-level approach for deriving the corrections was introduced. The focus was put on the absolute residual corrections using $Z(\rightarrow \mu\mu)+$jet events, as they are retrieved since the beginning of the CMS experiment by members of the IEKP for the Collaboration.

To guarantee fast determination and flexible studies of the correction factors for the latest accumulated data of the LHC Run II, the jet energy calibration suite EXCALIBUR was set up, based on the general event-based analysis framework ARTUS. The technical implementation and underlying physics motivation of all necessary steps of event and physics object manipulation and filtering for the calibration were described. For illustration purposes the determination of the calibration factors was shown using events recorded in 2015 with a center-of-mass energy of 13 TeV. This will help newcomers to understand the technical aspects of the calibration, as there are often discrepancies between physics methods and considerations and the way they implemented in software.

The EXCALIBUR suite was in-depth synchronized with the former calibration software of Run I of the LHC, the 8 TeV results were perfectly reproduced. The software was also successfully used to study the effects of the increased center-of-mass energy using simulated events at 8 TeV and 13 TeV, to help the jet energy calibration group with the preparations for LHC Run II. The new $Z(\rightarrow \mu\mu)+$jet calibration software has been in production since the start of Run II.
Chapter 8

Conclusion

With the increase in center-of-mass energy and luminosity in Run II, the LHC provides the opportunity to observe physics processes at unprecedented energy ranges and with high precision. While this offers new opportunities to physics analyses, the increased luminosity multiplies the already high rates of data recorded by LHC experiments and the necessary amount of simulated data.

This thesis successfully enabled the use of cloud-based virtualized resources providing additional computing capacities for physics simulations and analyses. In contrast to dedicated HEP data centers, virtualized cloud resources can be allocated on-demand, thereby reducing operating costs and maintenance effort while increasing the flexibility of resource provisioning. In addition, virtualization enables sharing of resources, e.g. classical HPC resources, between several research groups. A major part of this thesis was the successful integration of cloud resources located at the hybrid HPC cluster bwForCluster ENM at the University of Freiburg. The dynamic embedding of cloud resources into the local computing cluster was achieved by adding the necessary features to the cloud meta-scheduler ROCED and supplying a specialized virtual machine image. Access to this new resource is provided in a seamless and hassle-free way requiring no adaptations of existing HEP workflows or established job submission methods. During a 5 month period, various HEP workflows were successfully processed on a prototype of this new resource, using over 412,000 h of CPU time. The tools and methods developed as part of this thesis will allow a prompt utilization of the final cluster being installed in spring 2016. Furthermore, the contributions of this thesis allow future integration of additional cloud sites, e.g. from commercial providers, into the institute’s cluster.

To provide the CMS Collaboration with jet energy corrections for newly recorded data in a fast and flexible manner, the calibration software EXCALIBUR was set up for the new challenges of Run II as part of this thesis. It derives the absolute residual
corrections by analyzing $Z(\rightarrow \mu\mu)$+jet events where the $Z$ boson resembles a precisely measurable object to determine information about the jets. The new software successfully reproduced the corrections provided by the former software used for the LHC Run I and was used to study effects of the increased center-of-mass energy of the LHC during preparations for Run II. The new $Z(\rightarrow \mu\mu)$+jet calibration software has been in production since the beginning of data recording in 2015.
APPENDIX A

APPENDIX: ROCED LOG FILE

Listing A.1: Excerpt of a ROCED log file. Three management cycles are shown where new machines are requested on a cloud site and integrated into the batch system.
APPENDIX B

APPENDIX: JET ENERGY CORRECTIONS

B.1. Datasets

Json file used for the 2015 recorded data, released by the CMS Physics Validation: Cert_246908-260627_13TeV_PromptReco_Collisions15_25ns_JSON.txt.

Table B.1: Information about datasets and simulated events used in this thesis.

<table>
<thead>
<tr>
<th>Type/Run</th>
<th>Energy</th>
<th>Dataset</th>
<th>Global Tag</th>
<th># Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 2015C</td>
<td>13 TeV</td>
<td>/DoubleMuon/Run2015C-PromptReco-v1/AOD 74X_dataRun2_Prompt_v1</td>
<td>12194649</td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>13 TeV</td>
<td>/DYJetsToLL_M-50_TuneCUETP8M1_13TeV-amcatnloFXFX-pythia8/RunIIISpring15DR74-Asympt25ns_MCRUN2_74_V9-v3/AODSIM</td>
<td>28825132</td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>8 TeV</td>
<td>/DYJetsToLL_M-50_TuneZ2star_8TeV-amcatnlo-pythia8/RunIIISummer12DR53X_80Xv1-v1/Asympt25ns_MCRUN2_74_V9-v3/AODSIM</td>
<td>30458871</td>
<td></td>
</tr>
</tbody>
</table>

Table B.2: Information about the jet energy corrections used in this thesis.

<table>
<thead>
<tr>
<th>Type/Run</th>
<th>Energy</th>
<th>JEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>13 TeV</td>
<td>Summer15_25nsV6_DATA</td>
</tr>
<tr>
<td>Simulation</td>
<td>13 TeV</td>
<td>Summer15_25nsV6_MC</td>
</tr>
<tr>
<td>Simulation</td>
<td>8 TeV</td>
<td>Winter14_V8_MC</td>
</tr>
</tbody>
</table>
B.2. JEC Level Name Mapping

Table B.3.: Overview of different naming schemes of the JEC levels. The correction steps are used in this thesis for simplification, while the name tag of the correction files and pipeline suffix refer to the CMS and calibration jargon.

<table>
<thead>
<tr>
<th>Correction step</th>
<th>File name tag</th>
<th>Pipeline suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>uncorrected</td>
<td>(none)</td>
<td>(empty)</td>
</tr>
<tr>
<td>RC</td>
<td>L1RC</td>
<td>RC</td>
</tr>
<tr>
<td>1</td>
<td>L1FastJet</td>
<td>L1</td>
</tr>
<tr>
<td>2</td>
<td>L2Relative</td>
<td>L1L2L3</td>
</tr>
<tr>
<td>3</td>
<td>L2L3Residual</td>
<td>L1L2L3Res</td>
</tr>
</tbody>
</table>

B.3. Muon and Jet ID
Table B.4.: The muon identification criteria (MuonID) for Run II [62].

<table>
<thead>
<tr>
<th>Property</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loose ID</strong></td>
<td></td>
</tr>
<tr>
<td>Particle-Flow muon ID</td>
<td>true</td>
</tr>
<tr>
<td>Is global or tracker muon</td>
<td>true</td>
</tr>
<tr>
<td><strong>Medium ID</strong></td>
<td></td>
</tr>
<tr>
<td>Loose Muon ID (from above)</td>
<td>true</td>
</tr>
<tr>
<td>Fraction of valid tracker hits</td>
<td>&gt; 0.8</td>
</tr>
<tr>
<td>Good global muon</td>
<td>true</td>
</tr>
<tr>
<td>OR Tight segment compatibility</td>
<td>&gt; 0.451</td>
</tr>
<tr>
<td><strong>Tight ID</strong></td>
<td></td>
</tr>
<tr>
<td>Particle-Flow muon ID</td>
<td>true</td>
</tr>
<tr>
<td>Is global muon</td>
<td>true</td>
</tr>
<tr>
<td>$\chi^2/n_{\text{dof}}$ of the global-muon track fit</td>
<td>&lt; 10.0</td>
</tr>
<tr>
<td>Muon-chamber hits in global-muon track fit</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>Muon segments in muon stations</td>
<td>&gt; 1</td>
</tr>
<tr>
<td>Transversal distance ($d_{\text{z}}$) of tracker track and primary vertex</td>
<td>&lt; 2 mm</td>
</tr>
<tr>
<td>Longitudinal distance ($d_{\text{z}}$) of tracker track and primary vertex</td>
<td>&lt; 5 mm</td>
</tr>
<tr>
<td>Number of pixel hits</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>Number of tracker layer hits</td>
<td>&gt; 5</td>
</tr>
</tbody>
</table>

Table B.5.: The jet identification criteria (JetID) for Run II [63].

<table>
<thead>
<tr>
<th>Property</th>
<th>Loose ID</th>
<th>Tight ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>For $</td>
<td>\eta</td>
<td>\leq 3.0$</td>
</tr>
<tr>
<td>Neutral Hadron Fraction</td>
<td>&lt; 0.99</td>
<td>&lt; 0.90</td>
</tr>
<tr>
<td>Neutral EM Fraction</td>
<td>&lt; 0.99</td>
<td>&lt; 0.90</td>
</tr>
<tr>
<td>Number of Constituents</td>
<td>&gt; 1</td>
<td>&gt; 1</td>
</tr>
<tr>
<td>Additionally for $</td>
<td>\eta</td>
<td>\leq 2.4$</td>
</tr>
<tr>
<td>Charged Hadron Fraction</td>
<td>&gt; 0</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>Charged Multiplicity</td>
<td>&gt; 0</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>Charged EM Fraction</td>
<td>&lt; 0.99</td>
<td>&lt; 0.90</td>
</tr>
<tr>
<td>For $</td>
<td>\eta</td>
<td>&gt; 3.0$</td>
</tr>
<tr>
<td>Neutral EM Fraction</td>
<td>&lt; 0.90</td>
<td>&lt; 0.90</td>
</tr>
<tr>
<td>Number of Neutral Particles</td>
<td>&gt; 10</td>
<td>&gt; 10</td>
</tr>
</tbody>
</table>
B.4. Additional Plots of the 8/13 TeV Monte-Carlo Comparison

Figure B.1: Additional muon kinematics: $p_T^\mu$, $\eta^\mu$ and $\phi^\mu$. 
Figure B.2.: Additional jet kinematics: $p_T^{\text{jet}2}$, $\eta^{\text{jet}2}$ and $\phi^{\text{jet}}$.

Figure B.3.: $y^Z$ and $E_T^{\text{miss}}$. 
Figure B.4: $p_T$ balance (left) and MPF (right) jet response over the number of reconstructed vertices $n_{\text{PV}}$. Both responses are essentially flat, indicating a good pileup handling.


[37] “Microsoft OFFICE ONLINE”.
   https://products.office.com/de-deoffice-online/ (visited on 21/02/2016).


[42] “OPEN GRID SCHEDULER project website”.


(visited on 11/01/2016).

*Journal of Physics: Conference Series* 331 (2011), no. 6, 062034,

[46] T. Hauth, “Dynamic Extension of Local Batch Systems with Cloud Resources and
Measurement of the Jet Energy Scale with the CMS Detector”,. Diploma thesis, KIT

[47] S. Riedel, “Integration of Cloud Resources into Workflows of Particle Physics
Analyses and Measurement of the Underlying Event in Proton-Proton-Collisions

website”. https://github.com/roced-scheduler/ROCED (visited on
13/01/2016).

(visited on 25/01/2016).

[50] Karlsruher Institut für Technologie, “bwHPC C5 project website”.
https://www.bwhpc-c5.de/ (visited on 22/01/2016).

[51] Universität Stuttgart: BelWü-Koordination, “BelWü website”.
https://www.belwue.de/ (visited on 05/03/2016).

[52] University of Freiburg, “The testbed cluster NEMO”.
https://www.hpc.uni-freiburg.de/nemo (visited on 22/01/2016).

com/products/hpc-products/moab-hpc-suite-enterprise-edition/
(visited on 22/01/2016).


List of Figures

2.1. The accelerator chain at CERN. ................................. 5
2.2. A cutaway view of the CMS detector. .......................... 6
3.1. Users submit their computing jobs to a batch system queue whereupon they are distributed to free worker nodes. .......................... 11
4.1. The three delivery models of cloud computing. ................... 14
4.2. Virtualization allows separation of responsibilities. .......................... 17
5.1. Dynamic integration of cloud resources into the local computing cluster. .......................... 23
6.1. Picture of the racks for the forthcoming bwForCluster ENM. ........... 28
6.2. Interaction of tools and technologies for accessing cloud resources at the bwForCluster ENM. ........................................ 31
6.3. Resource allocation over time of a skimming workflow using cloud virtualized worker nodes on the bwForCluster. .......................... 34
6.4. Excerpt of a full CMS tt simulation workflow at 8 TeV center-of-mass energy illustrating draining machines .......................... 35
7.1. Schematic overview of the jet evolution and measurement. ............... 40
7.2. Multi-level jet energy correction approach of CMS. .................. 40
7.3. The ideal Z(\rightarrow \mu\mu)+jet topology: a single jet balancing the Z boson. .......................... 42
7.4. The modular structure of an ARTUS analysis. ........................ 46
7.5. Overview of all modules used for the Z(\rightarrow \mu\mu)+jet calibration. .......................... 47
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6</td>
<td>Z mass distribution of first LHC Run II data.</td>
<td>49</td>
</tr>
<tr>
<td>7.7</td>
<td>Distribution of transverse momentum and pseudorapidity of the leading muon with the corresponding selections indicated.</td>
<td>52</td>
</tr>
<tr>
<td>7.8</td>
<td>Distribution of transverse momentum and pseudorapidity of the leading jet with the corresponding selections indicated.</td>
<td>53</td>
</tr>
<tr>
<td>7.9</td>
<td>Distribution of $\Delta \phi$ of the leading jet and Z boson, $\alpha$ and transverse momentum of the Z boson with the corresponding selections indicated.</td>
<td>54</td>
</tr>
<tr>
<td>7.10</td>
<td>Absolute and relative selection efficiency of all selections on the number of events.</td>
<td>54</td>
</tr>
<tr>
<td>7.11</td>
<td>$p_T$ balance jet response.</td>
<td>55</td>
</tr>
<tr>
<td>7.12</td>
<td>MPF jet response.</td>
<td>56</td>
</tr>
<tr>
<td>7.13</td>
<td>Extrapolation of the jet response to the ideal event topology to retrieve the absolute residual corrections.</td>
<td>57</td>
</tr>
<tr>
<td>7.14</td>
<td>Number of pileup vertices before (left) and after (right) the reweighting.</td>
<td>58</td>
</tr>
<tr>
<td>7.15</td>
<td>Comparison of the leading muon transverse momentum and pseudorapidity at 8 and 13 TeV.</td>
<td>59</td>
</tr>
<tr>
<td>7.16</td>
<td>Z boson transverse momentum and mass distributions.</td>
<td>59</td>
</tr>
<tr>
<td>7.17</td>
<td>Comparison of leading jet’s transverse momentum and pseudorapidity at 8 and 13 TeV.</td>
<td>60</td>
</tr>
<tr>
<td>7.18</td>
<td>Transverse momentum of the second leading jet and $\alpha$ distribution.</td>
<td>61</td>
</tr>
<tr>
<td>7.19</td>
<td>$p_T$ balance and MPF jet response over $p_T^Z$ for 8 TeV and 13 TeV simulated events.</td>
<td>61</td>
</tr>
<tr>
<td>B.1</td>
<td>Additional muon kinematics: $p_T^{\mu}$, $\eta^{\mu}$ and $\phi^{\mu}$.</td>
<td>70</td>
</tr>
<tr>
<td>B.2</td>
<td>Additional jet kinematics: $p_T^{\text{jet}}$, $\eta^{\text{jet}}$ and $\phi^{\text{jet}}$.</td>
<td>71</td>
</tr>
<tr>
<td>B.3</td>
<td>$y^Z$ and $E_T^{\text{miss}}$.</td>
<td>71</td>
</tr>
<tr>
<td>B.4</td>
<td>$p_T$ balance and MPF jet response over $n_{PV}$ for 8 TeV and 13 TeV simulated events.</td>
<td>72</td>
</tr>
</tbody>
</table>
**List of Tables**

B.1. Information about datasets and simulated events used in this thesis. . . 67  
B.2. Information about the jet energy corrections used in this thesis. . . . . . 67  
B.3. Overview of different naming schemes of the JEC levels. . . . . . . . . . 68  
B.4. The muon identification criteria (MuonID) for Run II. . . . . . . . . . . . 69  
B.5. The jet identification criteria (JetID) for Run II. . . . . . . . . . . . . . 69