Monitoring and Optimization of ATLAS Tier 2 Center GoeGrid

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
„Doctor rerum naturalium“
der Georg-August-Universität Göttingen

im Promotionsprogramm ProPhys
der Georg-August University School of Science (GAUSS)

vorgelegt von

Erekle Magradze

aus Tbilisi

Göttingen, 2015

Betreuungsausschuss

Prof. Dr. Arnulf Quadt
Prof. Dr. Ramin Yahyapour

Mitglieder der Prüfungskommission:

Referent:	Prof. Dr. Arnulf Quadt II. Physikalisches Institut, Georg-August-Universität Göttingen
Korreferent:	Prof. Dr. Ramin Yahyapour Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen - GWDG

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Stan Lai
II. Physikalisches Institut, Georg-August-Universität Göttingen

Prof. Dr. Jens Grabowski
Institute of Computer Science, Georg-August-Universität Göttingen

Jun.-Prof. Dr. Steffen Schumann
II. Physikalisches Institut, Georg-August-Universität Göttingen

PD Dr. Ralf Bernhard
II. Physikalisches Institut, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 11.01.2016

To My Daughter Elene Magradze

Monitoring and Optimization of ATLAS Tier 2 Center GoeGrid

Abstract

The demand on computational and storage resources is growing along with the amount of information that needs to be processed and preserved. In order to ease the provisioning of the digital services to the growing number of consumers, more and more distributed computing systems and platforms are actively developed and employed. The building block of the distributed computing infrastructure are single computing centers, similar to the Worldwide LHC Computing Grid, Tier 2 centre GoeGrid. The main motivation of this thesis was the optimization of GoeGrid performance by efficient monitoring. The goal has been achieved by means of the GoeGrid monitoring information analysis. The data analysis approach was based on the adaptive-network-based fuzzy inference system (ANFIS) and machine learning algorithm such as Linear Support Vector Machine (SVM).
The main object of the research was the digital service, since availability, reliability and serviceability of the computing platform can be measured according to the constant and stable provisioning of the services. Due to the widely used concept of the service oriented architecture (SOA) for large computing facilities, in advance knowing of the service state as well as the quick and accurate detection of its disability allows to perform the proactive management of the computing facility. The proactive management is considered as a core component of the computing facility management automation concept, such as Autonomic Computing. Thus in time as well as in advance and accurate identification of the provided service status can be considered as a contribution to the computing facility management automation, which is directly related to the provisioning of the stable and reliable computing resources.
Based on the case studies, performed using the GoeGrid monitoring data, consideration of the approaches as generalized methods for the accurate and fast identification and prediction of the service status is reasonable. Simplicity and low consumption of the computing resources allow to consider the methods in the scope of the Autonomic Computing component.

Contents

1 Preface

Level of integration of the digital systems in different aspects of everyday life has increased dramatically during the last several decades. Rapid development of the internet, data storage and processing technologies and its growing consumption in scientific, commercial, medical and social spheres changed the scale of expectations and requirements to the digital service providers.
Failure or degradation of such services, even for several minutes may affect various aspects of life and causes economical and financial damage to both, the service consumers and providers. In case of scientific computing, disability of access to the computational and storage resources is a waste of CPU time and memory, which can be immediately converted into currency.
Modern large scale digital service providers have a complex, distributed structure and consist of a large number of heterogeneous hardware and software components. Most of these systems are based on the concepts of the Service Oriented Architecture (SOA). The SOA based systems provide the standardized communication protocols by means of services, which allow the communication between the services and not only with the service consumer. Thus different systems are able to autonomously communicate with each other and check their state in a given moment of time, which is used as basic information for making the decision about sending or aborting the next request. Ability of communication and information exchange between the heterogeneous systems motivated the development and evolution of the advanced automation tools for optimization of system administration and management tasks. Example of such a concept is Autonomic Computing. The concept considers coupling of each instance of a large computing infrastructure, i.e. hardware or virtual servers, network and power systems, with the Autonomic Manager. The purpose of Autonomic Manager is to collect the monitoring data from the instances and transform it into the knowledge about the infrastructure. The accumulated knowledge allows to perform the rapid, automatic problem detection and

its root cause analysis, hence optimizing the need on manpower resources. In addition, autonomic computing allows to perform the proactive management of the infrastructure, which assumes in advance identification of the state of a particular service or infrastructure component. Architecture of Autonomic Computing systems considers the task of monitoring information aggregation and analysis as a functionality of the dedicated component. Implementation and deployment of such a component is a hot topic of research and belongs to the area of efficient and highly available service provisioning, with the capabilities of automatic problem detection and solution. The core part of such an analytical component is the data processing algorithms.

The topic and the goal of this research directly corresponds to the key aspect of the Autonomic Computing. Thus, as a source of the monitoring information and the object for automation the Tier 2 centre of Worldwide LHC Computing Grid has been selected. The center is called GoeGrid and significant share of its resources is dedicated for the ATLAS experiment, which takes place at Large Hadron Collider at CERN[1]. The detailed overview of the LHC and the ATLAS experiment, as well as description of the scale of the demand on the computational and storage resources is provided in the second chapter of this thesis. Detailed overview of the evolution, as well as the key functionalities of Grid and Cloud computing platforms is discussed in chapter three, which also covers the description of structure and the organization of the distributed computing infrastructure for the ATLAS experiment. The detailed overview of GoeGrid hardware and storage resources, as well as its software setup is covered in chapter four. The fifth chapter is dedicated to the detailed description of the Autonomic Computing concept and its architecture. The next chapter consists of the description and overview of the mathematical foundation used for the monitoring data analysis. The seventh chapter provides the results of the conducted case studies and demonstrates the efficiency of the proposed approach for the quick and accurate service status identification and forecasting. The final chapter of this thesis concludes and summarizes the research.

Main motivation of this research was provisioning of the systematic approach for performance optimization of the GoeGrid resource centre. This goal has been achieved, however taking into account the generalized assessment of the case studies allows to consider this techniques as applicable for the similar, large scale computing facilities.

[1]European Organization for Nuclear Research.

2 The Large Hadron Collider

Human beings try to explain all events and processes ongoing in the surrounding world. The scientific research in various directions attempt to provide answers to these challenging questions for ages. These attempts lead to an evolution of fundamental sciences devoted to the explanation of the processes and regularities in the alive and material world.
Accumulated knowledge and experimental results allowed to propose the Standard Model of Particle Physics (SM), which is devoted to the explanation of the fundamental structure of matter and its properties. The SM was proposed in the 70s of the twentieth century and it describes what the observable universe is made of with few building blocks and is governed by four fundamental forces. There are certain limitations of the SM, however it is one of the main driving forces for the experimental and theoretical sides of the fundamental research in Physics.
This section is devoted to the overview and description of the Large Hadron Collider (LHC) and the ATLAS experiment, which is one of the main sources of the experimental data further stored and analysed on the Worldwide Large Hadron Collider Computing Grid (WLCG) [1], which consists of a complex infrastructure and the topic of its management automation remains active since the start of its operation.

2.1. Introduction

Organisation européenne pour la recherche nucléaire (CERN) is one of the few places in the world where the fundamental experimental research and validation of matter structure and its properties is conducted. The largest experimental facility at CERN and in the world is the LHC, which is a particle accelerator. Its length is 26.7 km and located approximately 120 meters in depth under the ground surface in the Franco-Swiss border area, near Geneva, Switzerland. The length and the scale of the LHC apparatus is exceptional among the existing accelerators and experiments. It allows to speed up the number of protons in opposite directions in order to collide them in the specially designed detectors placed in four locations of the accelerator circle. The set of accelerated particles is called bunches, the particles can be either protons or heavy ions, hence LHC can perform three types of particle collisions proton/proton, proton/ion and ion/ion. Each proton bunch consists about $115 \cdot 10^9$ particles, which are accelerated close to the speed of light allowing to reach the design value of the instantaneous luminosity of $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$. At the particle collision locations four detectors, ALICE, ATLAS, CMS and LHCb, (see fig. 2.1) are located, which serve as the main sources of the experimental data for analysis.

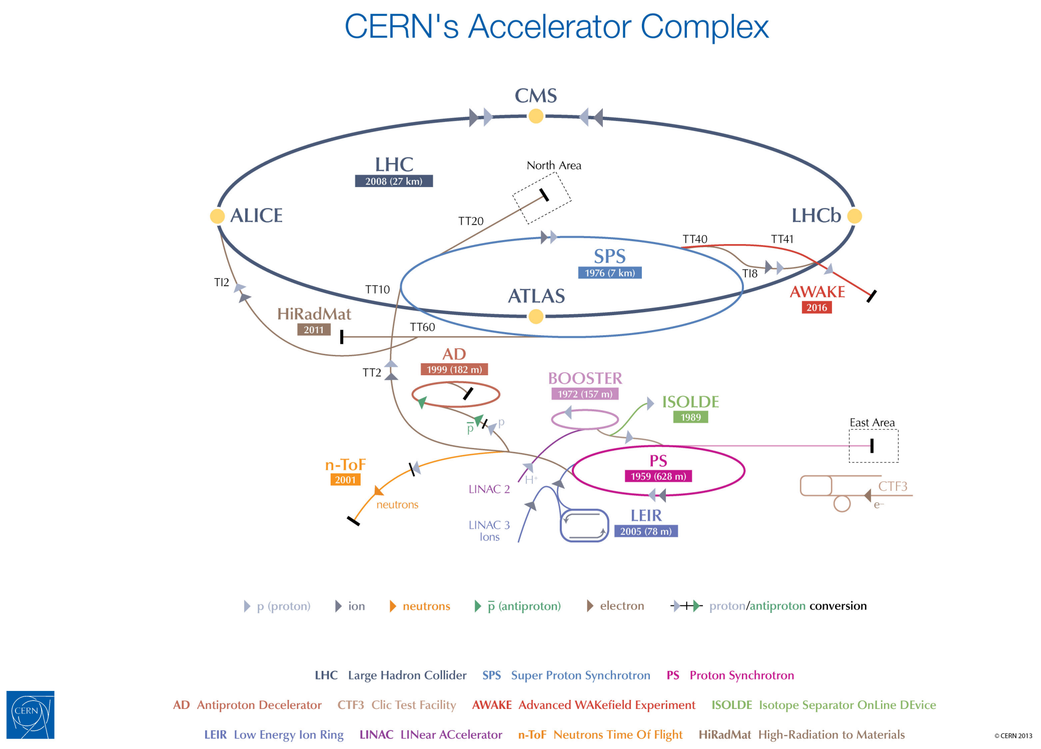


Figure 2.1.: The CERN Accelerator Complex ©CERN.

2.2. ATLAS Experiment

The largest and one of the multi-purpose detectors at LHC is the ATLAS (A Toroidal LHC Apparatus) (see fig. 2.2). Its length is 44 m and diameter equals to 25 m. It weighs about 7000 tons, consists of a large number of sub-detectors and is capable to handle 40 million events per second, which is sufficient to the bunch crossing rate of 25 ns.

Particles accelerated at LHC are colliding in four locations of the circle, where the

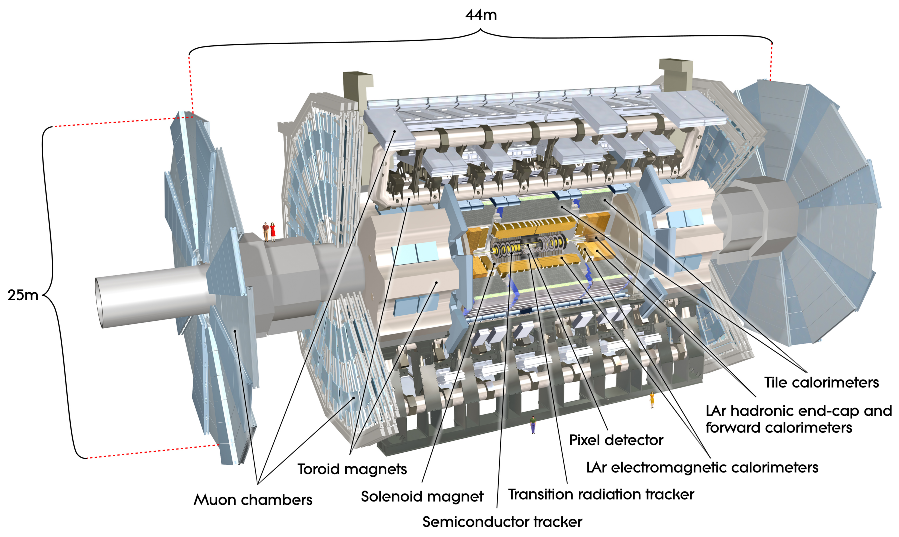

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Transition radiation tracker
Muon chambers
Solenoid magnet
Semiconductor tracker

Figure 2.2.: The ATLAS Detector ©CERN.

detectors are located. The particle bunches interaction in the ATLAS detector leaves the signatures of several primary vertices, which are considered the points of the particle collisions. As a results of the collisions, a large number of subatomic particles are generated and spread in all possible directions of the detector. Measurement of the full few-momentum of the particle and its different properties is the primary goal of the ATLAS detector and corresponding infrastructure.

Achievement of the primary goal is possible due to the multiple layers of the sub-detectors of ATLAS. The sub-detectors transform the physical event into the digital information allowing to investigate and observe the behaviour and properties of matter on high energies. Different parts of ATLAS are dedicated to the detection of a specific type of elementary particles. A general overview of the ATLAS constituents will be provided in the next paragraphs by finalizing with the descriptions of the experimental data flow from the detector to the storage system.

2.2.1. The Inner Detector

After the interaction of the bunches about 1000 particles, with different direction, charge and momentum are generated. The Inner Detector (ID), surrounds the vacuumized tube, where the elementary particles are travelling and colliding. It consists of three sub-detectors system. Due to the closest location to the primary vertex, the ID should handle large amounts of radiation, high temperature and a tremendous magnetic field,

generated by the surrounding solenoids and pervading all parts of ID. Therefore the design of such a detector was an extraordinary technological and construction challenge.
The first layer of ID sub-detectors passed by the generated particles after the proton bunches interaction is the Pixel Detector (PD). The PD consists of an array of silicon semiconductor sensors and may detect only the particles, which ionise the matter by means of electromagnetic interaction. Furthermore, magnetic field pervading the entire ID allows to trace the curvature of the generated particle paths, hence allows to reconstruct the particles charge and momentum.
The Pixel Detector is encircled by the Silicon Microstrip Tracker (SCT). The SCT, in contrast to the Pixel Detector, is able to detect the particles using the extruded, stereo silicon strips instead of pixels. The SCT is able to provide the information about the further path of the generated particles escaped from the PD using large number of read-out channels. The main purpose of the SCT is to give information about the charge and the momentum of the particle escaped from the PD.
The out-most layer of the ID system is the Transition Radiation Tracker (TRT), which provides the further coordinates of the particle path escaped from the SCT and allows to distinguish certain types of particles[1].

2.2.2. Calorimeters

The next set of sub-detector systems surrounding the inner detector is the calorimeters. The calorimeters are designed to identify the particle energy by transforming of their kinetic energy in another form of measurable energy. Two types of calorimeters, the electromagnetic and the hadronic calorimeter surround the ID.
There is a solenoid magnet in between the electromagnetic calorimeter (ECal) and the TRT. The ECal has sufficient size for tracking the relatively light particles, e.g. e^-, e^+ and γ, interacting on the electromagnetic level, hence providing the sufficient information about their energy.
Despite to the ECal the Hadron Calorimeters (HCal) is thicker and filled up with the strongly interacting material. The structure of the HCal allows the heavy particles, escaped from ECal, to produce the measurable decay products.

2.2.3. Muon Chambers

All the described layers of the ATLAS sub-detector systems are able to trace most the key sub-atomic particles but the muons. Due to the high mass of muons, they escape from all the described detector systems and leave their track in the outermost and largest part of the ATLAS, in the muon spectrometers.
The information provided from the outermost detector layer of the ATLAS is combined with the tracks from the rest of the sub-detector systems and makes it possible to select the few special collisions.

[1]electrons and pions.

Information provided by the set of detector systems is combined to get the final picture of the track of generated particles from the point of proton bunches interaction until they leave ATLAS.

2.2.4. ATLAS Raw Data Flow

The amount of data coming from the readout channels of all sub-detectors is estimated to be almost 60TB/sec, hence it is impossible to write the data with such rates in the storage system. Therefore, special data preprocessing steps are defined in order to reduce the data flow and extract only the useful information, worth for long term saving. This subsection is dedicated to the short overview of the data flow from the primary vertex to the storage infrastructure.

The proton bunch collisions rate can reach from 20 MHz to 40 MHz, hence after each collision enormous amounts of particles are emitted. Each of these emittance contain valuable experimental information and less useful data, which needs to be separated before writing in the storage system. Therefore about 90 million data channels of ATLAS detector, providing data streams mainly from the PD, are connected to the Trigger and Data Acquisition (TDAQ) system, which filters and sends to store only the valuable data.

The TDAQ system together with its supporting infrastructure, Detector Control System (DCS), guarantees the 1.6 MB of data stream per event. In total TDAQ system deals efficiently with about 2 million PB of raw data annually, which proves its efficiency and proper structure.

The TDAQ has three levels of the data filtering mechanisms, Level 1 and Level 2 triggers and the Event Filter (EF). The later two level systems form the High Level Trigger (HLT) system (see fig. 2.3).

The Level 1 trigger is implemented on hardware level and consists of a large number of parallel processors (ASIC FPGA), which filter all the events by making decisions about writing the data in every 2.5 microseconds. The data rate after L1 trigger drops down to 100.000 events/sec.

The Level 2 trigger is a software implementation and uses around 1000 CPU cores in order to analyses the data and keep or drop an event. The Level 2 trigger uses information from all ATLAS detectors and inspects the selection from the L1 trigger. Thus, after filtering of the data via the L2 trigger the event rate is dropped down to 3.5 KHz, with around 40 ms of average processing time per event.

The EF, i.e. the final level of the trigger system analyse the data emitted from the L2 trigger system using the software with implementations of complex mathematical calculations for the basic event reconstruction. Hence, the time spent for the analysis per event, reaches 4 seconds. The final reduction of event rate after the EF level varies from 200 to 400 Hz, which is the rate of the data writing in the storage system, i.e. it is about 320 MB/s or 3.2 PB per year. The overall reduction factor of the trigger system reaches about 180000.

The very final step of the experimental data flow, is the provisioning of the filtered data

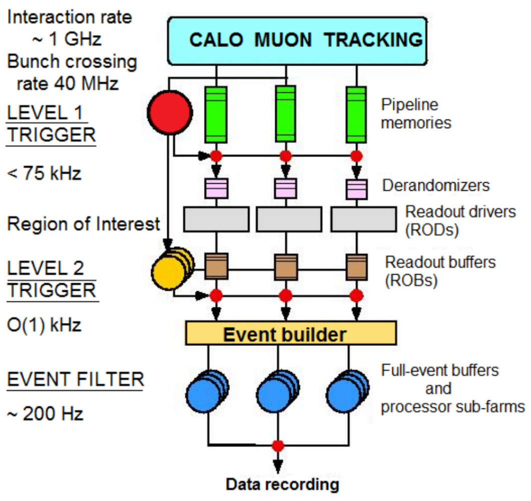


Figure 2.3.: The ATLAS Trigger System [2].

in a suitable format for further analysis.

3 Grid Computing

The formation and development of the Grid computing concept was motivated by several factors such as growing demand on the computational resources, increasing scale of the scientific and industrial challenges and the rapid development of computer network technologies.
Predecessors of the Grid computing concept were the large scale computing systems. In 1960s it was a common practice to prepare the code and the input data and wait in a queue to access the computing facility. Alongside with the shortage in terms of advanced slot reservation for the computing facilities, sometimes it was taking several days to get a result from the computer. The computing task taking data for the processing and giving the results was called the job.
Because of high price and the large service costs, computers were affordable only for a small number of organizations, hence there were not many users of such facilities. Due to the small demand on computing facilities, hardware limitations and specific not user-friendly communication interfaces, development of the computer technologies in 1960s was relatively slow. The situation has changed in 1970s when computers were first linked via networks, this technology eventually led to the formation of the ARPANET [3] the predecessor of the internet [4]. Since the appearance of the computer network the idea of remote access to the computing facilities has appeared and the number of consumers of digital services has started to grow in parallel with the rapid evolution of the corresponding hardware and software systems. In 1990s the scales of the internet and the technological breakthrough was sufficient for the further evolution of the concept of the distributed computing, which appeared in parallel with the development of the computer networks. The idea was to discover and use idle computers connected among a certain network. The first significant success in development of the distributed computing concept was achieved at the Xerox Palo Alto Research Center (PARC), where scientists John F. Shoch and Jon A. Hupp [5] developed the so-called "worm" application, which

was able to move through the computer network and discover idle resources.
Besides the computer network and related technologies evolution, the driving force for the development of the computing on demand i.e. the Grid concept was the significant change in industry and science. It was clear that for the next step in the research fields like climate change, High Energy Physics, Cancer and Tumour disease research, Seismology, Finances and Economics a handful of computers are not enough, even if these computing facilities are so-called super computers. Hence, in the beginning of 1990s inspired by the availability of high-speed computer networks, as well as by the sufficient number of the computing devices and by the challenges by the computational requirements of new applications the I-WAY (Information Wide Area Year) [6] project started. In terms of the I-WAY project several high-performance computers and advanced visualization environments were interconnected by an experimental, high-bandwidth computer network. This attempt of computing resources consolidation for a certain computing intensive applications turned out to be the precursor of the Globus Toolkit and the National Technology Grid, which later became the de-facto standard for a number of grid applications.
Nowadays, architecture and the organization of the Grid systems has changed a lot from times of I-WAY. The aim of this chapter is to provide the definition of the grid concept and describe how it differs from other later appeared technologies like cloud computing platforms [7], Peer-2-Peer systems [8] or Desktop Grids [9]. Architecture and description of the main components of the Grid system as well as the overview of the largest grid - WLCG components in physics is also provided in the next sections and sub-sections.

3.1. Definition

In the middle of 1990s, when the Grid initiative was in its very beginning, it was hard to provide the clear and generic definition of what grid computing is. A number of key components were changing from time to time, but the most important factor was the software connecting the computing resources. During the relatively short period of time, among the communities involved in I-WAY project the I-SOFT [10] infrastructure development has been spurred, which was the precursor of the first software meant for the functionality of the grid infrastructure. At the same time a large number of scientific communities were actively considering ways of the computing resource federation and this process was facilitated and continued into several Grid development projects mainly in the USA and Europe. These projects, e.g. EU DataGrid [11], Particle Physics Grid (ppdg.net) and Grid Physics Network [12] led to the creation of large distributed computing and data storage Grids, like Open Science Grid in the USA and EGEE, EGI in Europe and the international LHC Computing Grid (LCG) [13].

Nowadays, after the middle of 1990s, i.e. after the beginning of the "Grid era" we can identify many concepts, definitions and publications describing computational Grid, data Grid, knowledge Grid, discovery Grid, desktop Grid, cluster Grid, enterprise Grid, global Grid etc. Among the variety of Grid systems it is hard to identify common features and to provide a generic definition of the Grid, but according to Ian Foster's and

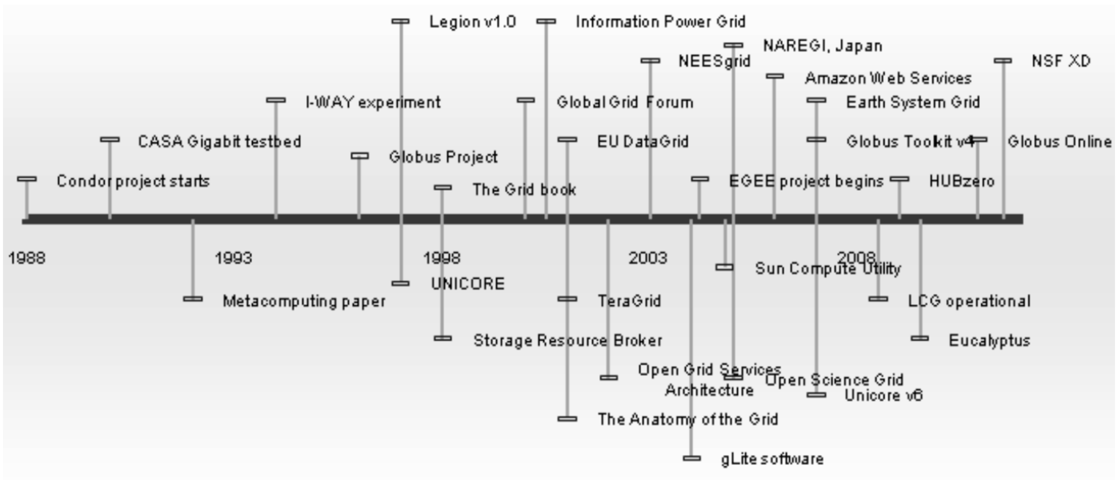


Figure 3.1.: 30 Grid and Grid-like systems in the period of 1988-2011.

Carl Kesselman's fundamental work "The Grid: Blueprint for a new computing infrastructure" [14].
"Computational Grid is a hardware and software infrastructure that provides dependable, pervasive and inexpensive access to high-end computational capabilities".
This definition came up when the formation of the Grid concept was still ongoing, therefore was not enough precise. Soon in 2001, this definition was altered in the next fundamental work "Anatomy of the Grid" [15], where the authors underlined the organizational aspect of a Grid:
The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.
This definition is rather complete compared to previous ones and removes borders among data or computing-oriented types of a Grid. In addition, based on the second definition, it is clear that the Grid has a complex layered structure and there are clear differences in a Grid infrastructure and a mere cluster of networked computers. The main difference is in the requirement of the administrative and organizational agreement of the sharing of the resources. In case of Grid, the resources has broader meaning and this term covers not only CPU power or memory resources but also data storage, network, information systems and credential data systems.
Nowadays numerous commercial or scientific computing and storage infrastructures employing distributed computing architecture, but these systems can hardly be identified as Grids. For clear separation of a Grid system from the cloud computing or SAAS (Software As A Platform) frameworks the three point check-list was proposed by Ian Foster [16]:

"*coordinates resources that are not subject to centralized control...*
A Grid integrates and coordinates resources and users that live within different control domains - for example, the users desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.
...using standard, open, general-purpose protocols & interfaces...
A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. As I discuss further below, it is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application-specific system.
...to deliver non-trivial qualities of service.
A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts."
This check list shapes the concept of the Grid systems more precisely and provides an understanding of three key points. Grid systems should operate without centralized point of control, which allows to dynamically organize the groups or collaborations demanding the computing resources. Once these groups are created the users and resources can be assigned to it and hence be authorised to operate the Grid system. Dynamic management and operation of such kind of systems would be impossible without the adoption of certain standards, open and general purpose protocols and interfaces. All of these makes it possible to integrate the wide range of heterogeneous hardware and software platforms under a single hood, which also means that the high level of interoperability is achievable.
After applying of these two points to the distributed computing system the third point can be achieved as a result. Delivery of a non-trivial Quality of Service (QoS) is the goal that needs to be achieved by the proper implementation of the Grid key points, which includes efficient usage of the available resources across multiple domains, which otherwise might be idle.
Alongside with the aforementioned key points there are three main characteristics that allows to separate Grid and other distributed computing systems [17].
"*Heterogeneity*: a grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.
Scalability: a grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.
Dynamicity or Adaptability: in a grid, a resource failure is the rule, not an exception. In fact, with so many resources in a Grid, the probability of some resource failure is

naturally high. The resource managers or applications must tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services."

3.2. Overview

All large scale and distributed systems have a complex structure, a Grid is not an exception. In the following subsections, the principles of Grid functionality architecture and the key components of a Grid system will be described.

3.2.1. Architecture

The Grid has a layered structure as shown in figure (see fig. 3.2), where the functionality of the upper layers is guaranteed by the proper functionality of each of the lower layers. The principles of the Grid architecture also refers to the "hourglass model" [18] where the neck plays a key role - connecting a massive amount of applications to the large number of computing resources by means of a limited number of key protocols.
The goal of this section is to describe each of the components shown in the layered architecture.

Fabric

The Grid infrastructure is formed by the number of large computing facilities spread aroud the world, which are accessible via the internet. These centers are formed on the basis of the heterogeneous hardware and software systems. This variety of systems i.e. interconnecting infrastructure, distributed file systems, computer clusters, workstations, storage systems form the computing resources of the Grid and are considered to be logical rather than the precise entities. These logical entities with different functionality and purpose due to the efficient implementation of the upper layers are used by the upper, Resource layer.
The key point for the computing resources provided by the Fabric layer is their efficient usage. In order to guarantee this, there are implemented the resource management systems in the upper layers of the Grid infrastructure. The "resource managers" communicate with the self-analysis systems implemented on the level of Fabric layer, which can be considered as another important component of this layer besides the hardware constituent.

Connectivity

Two main functionalities are provided by the connectivity layer: the communication and the user and resource authentication. The communication protocols allow data exchange among the number of computing and storage systems being part of the Fabric layer.

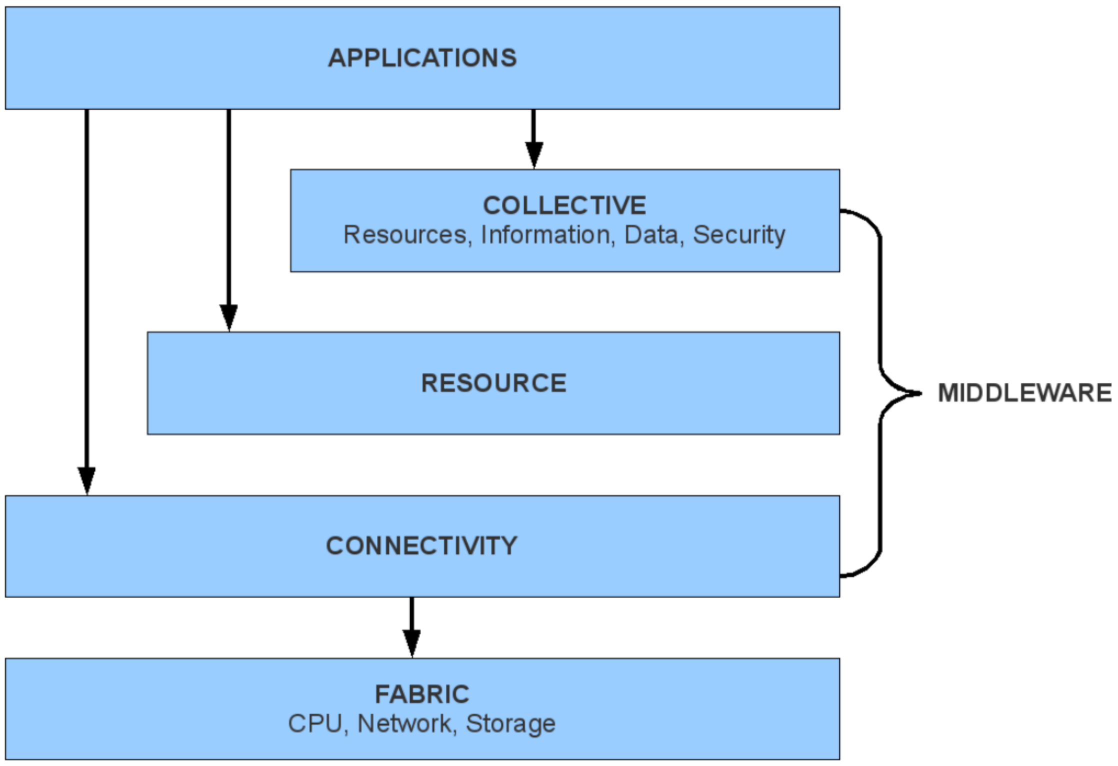


Figure 3.2.: Layered Architecture of the Grid [14].

The authentication mechanism in the Grid is meant to provide a transparent access to the resources, which is guaranteed by a single sign-on authentication mechanism.
Communication via the connectivity layer is organized by means of the secure protocols and the goal is to allow the consumption of the computing resources of the Fabric layer via certain protected interfaces.

Resource

In order to interact with the computing resources spread around the world, the resource layer provides a small set of protocols to the Grid users and applications. These protocols allow to establish a secure connection with the individual resources and also initiate, control and monitoring of the shared computing environment of the Fabric layers. The secure communication of the user with the Fabric layer is organized via the connectivity layer.
The protocols provided by the resource layer can be grouped in two classes: the information and the management protocols. Information protocols are meant for the state monitoring of the computing resources, also they allow to collect instant information about the status of the user activity. The management protocols are meant for the initiation and control of the shared environment in order to achieve the desired QoS.

Collective

The aim of the Collective layer is to define the protocols for the coordination and efficient usage of the distributed computing resources.

Applications

The Application layer is the point where the consumers of the Grid computing resources interact with all of the Grid layers. The interaction is possible via appropriate APIs, which are implemented in the Grid middleware. The Grid middleware is the specialized software for functioning and interoperability of the entire Grid system as a single computing entity. The APIs are useful not only for the Grid users but also for the developers since it makes possible to create an application that can take advantage of the resources available within the Grid.

3.2.2. Virtual Organization

Computing centers being part of the Grid infrastructure are spread around the world in a number of scientific institutions or private companies. The users of the Grid system are also from different countries and institutions, but are affiliated with certain groups or collaborations in order to reach a common goal. Users and the computational resources consolidated to reach a common goal, form so called Virtual Organizations (VO) [15], which is a core concept of the Grid.
Members and the computing resources of VO share certain rules, which might strongly differ among VOs. Despite these differences there are common principles, mainly related to the resource sharing rules within the Virtual Organization. These principles need to be followed when a new VO is created.
Resource sharing is regulated by the different constraints like the time and periods of usage, as well as the purpose and the type of the activities that is planned to take place. Besides the constraints from the resource provider side, there can be defined limitations also from the consumers part, like usage of the computing resources with the specific number of CPU cores. This set of limitations can be defined by means of the system policies in order to identify the user or the resource (authentication) as well as to find out what kind of activities are permitted for this particular entity (authorisation).
Members and the resources assigned to VO are not static and are considered to be subject to dynamic changes. The changes mainly refer to the number of VO members, which might increase or decrease during a short period of time. The resources are also subject of changes, which can be modified, extended or reduced. Tracking of such changes can be considered as an important requirement for the VO system implementations.
Another aspect of SLA[1]'s sharing is the resource combination and coordination, which means that, certain parts of a specific computational task can run on a particular hardware, while the rest of it can be redirected to another available resources. From this point of view the resource sharing relationships within the VO can be considered as a

[1]Service Level Agreement.

peer to peer communication i.e. resource provider can also be the resource consumer and vice versa.
The key requirement for a VO resources is the flexibility, i.e. the same computational resource can be used for a very specific task, while in other case it should be considered as a means for a generic compute cycle. All this information should be dynamically available in a VO system, which makes it possible to define general QoS[2] related requirements for the resources within a single VO.
These conditions related to the resource sharing are mainly considered as a general principle and characteristics for the Grid basic concept, the VO.

3.2.3. Components of the Grid Computing

Alongside with the core components of the Grid architecture there are a number of software systems that guarantee the coordinated and efficient integration and performance of all Grid layers. These software systems have changed and evolved since their creation but the key aspects remain the same.
The Grid concept is wide enough to use different technologies and approaches for its component implementations. However, this fact influences the interoperability of different Grid services among the computing centers i.e. the idea of the resource federation and sharing might not be realized efficiently or at all. This fact, as well as the success of the internet standards implementation and similarity of the Grid and the internet in terms of the resource federation, emerged a discussion in 1999 about the development and application of the Grid standards. Initially discussions were ongoing in terms of the Grid Forum, which later evolved to the Global Grid Forum and then to the Open Grid Forum (OGF).
Efforts among the OGF as well as within the other related organizations like Organisation for the Advancement of Structured Information Standards (OASIS) and the Internet Engineering Task Force (IETF) led to several successful standards development and deployment like: the proxy certificate profile [19], GridFTP [20], which extends the File Transfer Protocol for the Grid systems mainly for the scientific and research communities. Besides these examples there were efforts for the standardization of the Storage Resource Manager (SRM) specifications as well as for the related policy specifications [21].
Another example of the successful standardization effort is the Grid Laboratory for a Uniform Environment (GLUE) specification [22], which mainly allowed and extensively pushed forward the standardisation of the federation of task execution systems especially in WLCG.
Other attempts for the standardization of the Grid infrastructure aimed at more global goals e.g. the Open Grid Services Architecture (OGSA) [23] with Open Grid Services Infrastructure (OGSI) [24], which was substituted with the Web Services Resource Framework (WSRF) [25] in 2004.
The goal in case of the OGSA was the development of common standards and interfaces for the management of Grid resources. The final aim was to facilitate the interoperability

[2]Quality of Services

among the various Grid services. The OGSI was the infrastructure for the realization of the OGSA, which was mainly based on Web Service Definition Language (WSDL) and XML. Due to several issues related to the too high level of the object orientation of OGSI and also problematic functionality with the Web Services, it was substituted with the WSRF. The WSRF functionality mainly is the same as for the OGSI but it is decomposed into six standards based on the existing Web Service standards. Moreover it split the stateless Web Service from the stateful Grid resources, which are packaged by a Web services.

Security

The Grid offers not only the computational resources but it also provides services to store the data. Usually, for both, commercial and scientific entities who use the Grid infrastructure, permission of work with the specific data is a matter of concern. The good example for that is the high energy physics community, dealing with the extremely meaningful experimental data being analysed at the WLCG computing centers. In this case it is crucial to keep several security aspects for the WLCG resource consumers and providers, which does not differ for other Grid systems.
The key role in the secure usage of the Grid resources plays user and resource authentication and authorization. The authentication mechanism allows to identify and allow the user to the Grid resources and data stored in it, while it is also able to deny the access to the Grid infrastructure. The authorization mechanism defines the set of actions that user or an automatic entity, communicating with the Grid services, is allowed or restricted to perform.
Confidentiality alongside with the trackability are also important aspects in the Grid security system. The confidentiality isolates the resources and the data which is allowed to be consumed by the user or an automatic entity and also it guarantees that the user actions will not be published to the unauthorized third parties. The trackability guarantees that all actions performed by the user will be stored and in case of the violation of the security agreements by the Grid resource consumer, this information will be used to identify the culprit. The trackability together with the agreement on non-repudiation guarantees that the source of the security problem will not be able to deny involvement in the accident.
Data integrity together with the aforementioned aspects is also a key requirement to guarantee the secure environment in the Grid system, though it is specific for the HEP community. Experimental data produced at the LHC experiments is unique and each bit of this data has a high price, thus it is crucial to avoid any changes in the data and also it cannot be lost.
The example of the security system covering all the aspects is the Grid Security Infrastructure (GSI) a part of the Globus Toolkit. The GSI is the certificate-based system, which provides the X.509 standard based digital certificates to the Grid users and resource hosts. In this way the authentication and the authorization functionality for the Grid related entities is guaranteed.
When using the Grid infrastructure one need to retrieve the data from the storage sys-

tem, process it using the computing system and retrieved output needs to be saved back into the storage infrastructure on a special place. For all these general steps the authentication and authorization is required for each of the systems. The certificate-based mechanism allows to perform this action with the single-sign on functionality, which is possible due to the existence of the proxy certificate concept. The proxy certificate usually has the short life time, which can vary from twelve to one full day. All the user or host privileges encrypted in the main certificate are delegated to the proxy one and after that all the actions, for a particular task, requiring authentication and authorization, in different systems, is performed via the proxy certificate without requesting again and again the password or pass phrase. The proxy certificate mechanism also guarantees the confidentiality, traceability and non-repudiation aspects of the security. On the level of the automatic data distribution management the data integrity is also guaranteed based on the resource hosts digital certificates.

Workflow and Resource Management Systems

Advantage of the Grid technologies is its ability to serve for the multitude of tasks coming from different scientific or commercial groups or large collaborations. While the Grid security or resource usage systems are similar there are differences in the set and the sequence of steps that need to be taken for the solution of a particular problem. The Grid system provides mechanism to define the sequence and an order of steps for the Grid services usage to achieve the successful task accomplishment. This mechanism is called the workflow management system [26].
The workflow management systems mainly provide functionality to build the model for the definition of the steps in order to achieve the successful and efficient task execution. The model building process is related to the interoperability issues, which directly affect the efficiency of the resource federation idea in the Grid.
Though the workflow management system is one of the key parts among the Grid systems, it has a complex structure and consists of multiple substructures. According to [26] there are five core components of a general workflow management system: The Workflow design, Information Retrieval, Workflow Scheduling, Fault Tolerance and the Data Movement.
Systems belonging to each of these substructures can be classified into several groups.

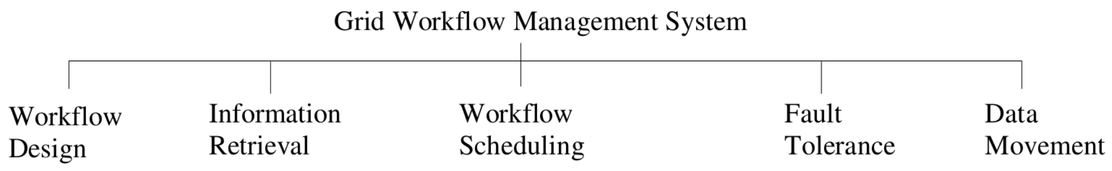


Figure 3.3.: Components of the Grid Workflow Management System [26].

Since the workflow management aspect is tightly related to the high efficiency of the computing facility, all the subcomponents of the above mentioned subsystems will be

described in general.
Each of the workflow management system has its own structure, according to the available systems these systems can be grouped in two classes according to their structure characteristics. These classes are the Direct Acyclic Graph (DAG) based and non-DAG based ones. Both of these structures are classified into four main groups: *Sequence*, *Parallelism*, *Choice* and *Iteration* [26].Difference between the DAG and non-DAG structures are the systems with the *Iteration* structure, that allows to run repetitive computational cycles. The rest of the subclasses of the structures are common for DAG and non-DAG oriented workflows.
The *Sequence* represents the workflows that run the task execution steps only sequentially i.e. without completion of the current step it is impossible to go on the next step of the task execution. The *Parallelism* considers possibility of running several steps of the workflow simultaneously. The *Choise* structure depends on the specific conditions, which are mainly defined by the user i.e. task owner. Due to the certain level of freedom provided to the users by the *Choise* structure, mostly in this case the workflows are the hybrid systems i.e. mixture of the *Sequence*, the *Parallelism* and the *Iteration* in case of the non-DAG approach.
Besides the structures it is important to define the model of the workflow management system in order to reach the maximum possible throughput from the infrastructure. According to the taxonomy provided in [26] there are abstract and concrete types of models. The type of the model depends on a way of the resource specification in the workflow. The abstract type of the models does not require the precise resource specification and a general statement of the attributes like, number of cores as well as the amount of memory and the free space on the disk, is sufficient. The workflow systems with the concrete model require detailed specification of the required resources, hence prior knowledge of the Grid fabric layer, the location of the data, which needs to be analysed. Though with the concrete workflow models the efficiency of the task execution can be increased dramatically still it is cumbersome for the users to clarify all the current details and specifications of the resources that needs to be involved in a particular sequence of tasks.
In order to implement different types of the workflow structures and models it is necessary to have a sufficient tool. The workflow composition systems are meant for this purpose. According to [26] there can be identified two main groups of the workflow composition systems the *User-directed* and the *Automatic*. The *User-directed* composition systems are mostly used by the experienced Grid users, who are able to detect the problem during the task execution and alter the workflow directly. In case of the *Automatic* composition systems the workflow design is generated automatically from the available options.
There are two ways of the workflow structure definition the language-based and the graph-based. The language-based approach uses the Extensible Markup Language (XML) [27], while the graph-based modelling is more intuitive and easy-to-use due to its Graphical User Interface (GUI). Examples of the graph-based systems are Petri Nets [28] and UML (Unified Modeling Language) [29].

On the step of workflow design it is important to allow users of the Grid infrastructure to define the structure and the model of the steps that needs to be done to achieve the desired result from the task execution. The challenging point is how to define what is a desired result. For that, at the step of workflow design it is possible to define workflow QoS constraints. These constraints might be related to the execution time limits, on the task and the workflow level as well. Examples of the systems allowing to implement such kind of constraints are UD, ED, EQS and EQF [30].
The Grid workflow management system aims to tie up the available computational resources to the task execution. This functionality would be impossible without collection of the information about the state of the Grid infrastructure components, ongoing processes in it, accessibility and availability details. Hence the information retrieval is one of the core components of the workflow management systems. This component mainly retrieves three types of the information from the Grid, the static, historical and dynamic [26].
The static information consists of the hardware characteristics and the configuration details of the Grid fabric layer e.g. operating systems, network bandwidth or the available software tools. Also it covers the user's authentication and authorization information together with the resource consumption cost information. The static information helps to plan the workflow better in order to achieve the high QoS and efficient performance.
The Grid infrastructure is meant for the shared usage, therefore the information about available resources, the network load and the accessibility to the services is a matter of active changes. Therefore this information can be categorized as dynamic. The dynamic information is useful in different workflow scenarios where the on fly modifications are possible.
In order to preserve the Grid usage experience of multiple users the historical information of the infrastructure needs to be aggregated. This information can be used in future to identify and tag resources according to the different categories like stability, capacity, reliability and serviceability.
Data aggregated and analysed by the information retrieval component of the Grid workflow management system can be used for efficiency and QoS increase. Examples of such components are Monitoring and Discovery System (MDS) [31] as well as Network Weather Service (NWS) [32].
The main subject of interest in terms of the Grid systems is the efficient usage of the data storage and computing resource management systems. Execution of a particular task consists of several general steps, like data retrieval from the data storage system, processing it, which requires communication with the computing resource management systems and finally collection of the results, which again means interaction with the data transfer and storage services. These steps need to be defined in the workflow management system. The success of the workflow depends on an efficient accomplishment of each of the steps. In order to deal with the multitude of the requests going to the various system the workflow scheduling component was introduced.
According to [26], different workflow scheduling systems have several key components that can also be used to distinguish these systems from each other. These components

are: *scheduling architecture*, *decision making*, *planning scheme*, *scheduling strategy* and *performance estimation.*
Several aspects like autonomy, performance, quality and scalability of the workflow depend on the scheduling architecture [33]. Approaches provided for the workflow scheduling can be classified in three main groups, centralized, hierarchical and decentralized.
The centralized scheduling architecture is efficient for the small scale workflows, where the number of the essential steps for the successful task accomplishment is law. Another feature of the centralized scheduling is the aggregation of the information about the resources at a single point and in this way control the entire workflow efficiency from a single point. This feature can be also considered as a weak point of such an approach, since problems at this single point can cause the disability of entire workflow.
Despite to the centralized scheduler the hierarchical and decentralized architectures consist of several sub-schedulers. These architectures are efficient and easily scalable for large Grid systems, where the task accomplishment requires big number of sequential steps. In case of the hierarchical architecture, there is a central, top-level scheduler which orchestrates certain number of sub-schedulers. The sub-schedulers manage the certain share of the computing resources and interoperate with each-other and the central scheduler, which checks and leads the steps of the workflow execution process among the sub-schedulers. The example of the example of the hierarchical scheduler is Grid-Flow project [34], which consists of a single workflow manager and the number of low level schedulers.
The decentralized option of the workflow scheduling architecture consists of several schedulers without any subordination i.e. there is no central and dependent schedulers. For such schedulers the key aspect is an interoperability of the components.
Though, hierarchical and decentralized schedulers are suitable for the large workflows with multiple steps, there are several problems that might arise during the usage of such systems. The challenging in case of the hierarchical scheduler is the stability of the central manager, its malfunctioning can cause the failure of the sub-schedulers and the entire workflow. For the decentralized architecture the main problem is the risk of conflicts i.e. in case of the poor interoperability among the schedulers there might occur competition for the same computational resources, which could server as a reason for stopping of the task execution process.
Together with the scheduler architecture the important aspect is the proper decision making when mapping the available resources and tasks. The resource and the task tie up process can be done on a task and on the workflow levels [35].
In case of local decision i.e. task level decision making process the resources can be maped to the task in terms of the sub-workflow and as a result the high efficiency can be reached, but the local decision might be harmful for the entire workflow and there is a risk of conflicts in terms of the resource consumption. Due to this reason the local decision making approach is not used in small computing infrastructures where the shortage of the resources can be considered as an issue of QoS.
The global decision making, i.e. the decision making process on the workflow level aims to provide the best possible performance of the entire task execution process, but the

number of parameters that needs to be taken into account is high. Hence the global decision making in terms of scheduling can take long time and decrease the overall workflow efficiency. When relying on the global decision making approach it is important to find the trade off between the workflow efficiency and the time required for the scheduling of the task execution steps.
The workflow scheduling requires the good prior knowledge of the information about the hardware characteristics of the Grid fabric layer as well as the information about the state of the resources and ongoing processes in it. For the scheduling component of the workflow management system this information is used for the planning of the task execution steps.
According to the [26] there are two classes of the planning approaches, the static and the dynamic. The workflow management system with the static planning relies on the static information, which contains the information remaining unchanged for the relatively long period of time, mainly this is the characteristics of the fabric layer. The schedulers with the dynamic planning take into account the static and the actual information about the service availability, serviceability and accessibility. This information is analysed and the scheduling plan is defined, which can require the workflow change during the task execution.
Each of the classes of the workflow scheduling components contain several subclasses. In case of the static planning there are identified the user-directed and the simulation-based strategies [26]. The user-directed planing strategy depends on the Grid user experience and the knowledge of the fabric layer, while schedulers with simulation-based planning runs several test tasks in the Grid environment in order to identify the reliable and suitable resources. Information retrieved from the test task executions serves as a basis for the dynamic planing strategies to define the most reliable and efficient workflow. General concept of the dynamic planning allowed to develop certain number of workflow management sub-systems, which are also classified in two groups the prediction-based and the just-in-time.
The prediction-based planning of the workflow is a lot similar to the simulation-based approach but in case of the prediction-based planing strategy it is possible to change the workflow schedule during the task execution i.e. on fly. The disadvantage of the prediction-based scheduler is high sensitivity to the prediction accuracy.
The just-in-time planning of the scheduling strategy allows to define the sequential or parallel execution of the steps of the tasks just before launching of its execution [36]. This kind of workflow planning is efficient in homogeneous Grid environments but in case of the dynamic and heterogeneous Grid systems, where the better resource can appear any-time, the just-in-time planning can lead to the lower efficiency and to the less optimal decisions in terms of the resource usage.
Alongside with the scheduler architecture, decision making strategy and the planning scheme it is important to define the scheduling strategy for the scheduling component of the workflow management system. Mostly, workflow scheduling strategies depend on a specific task and the related QoS constraints but still it is possible to generalize these methods to the performance-driven, trust-driven and market-driven classes.

The performance-driven scheduling method aims to organize the intermediate workflow steps in order to optimize the execution time. Examples of such systems are GrADS [37] as well as the scheduling scheme described by Prodan et.al [38].
Among the number of metrics characterizing the workflow efficiency the important role plays the reliability of the resources. Besides certain security attributes that are required to the computational resources for reliability, it is important also to have the statistics with low number of failures per the resource unit. The reliability information extracted from the performance statistics serves as a basis for the trust-driven strategy. Methods of the trust-driven scheduling applies certain ranking to the resources with higher reputation in terms of the large share of the successfully accomplished tasks.
Unit resource consumed in Grid can be translated into certain amount of financial resources and hence to optimize the cost per task execution is the goal of the market-driven scheduling strategy. Besides the cost the strategy also aims to meet the user requirements in terms of time limits per task execution.This approach is implemented in a number of software systems like Nimro-G [39] and Gridbus data resource broker [40].
One of the key aspects in the scheduling systems is the estimation of the resources and time consumption by the task. According to the [26] there are five groups of methods dedicated to the task execution performance assessment. Mainly the methods differ from each other according to the ways of the performance information acquisition.
The simulation method [41], [42] is based on a pretesting of the task execution workflow in a specialized simulation environments that mimic the real Grid infrastructure. The analytical modelling method [43], [37], [44] takes into account the available resource characteristics alongside with the analytical measurements and based on task specification predicts the its execution performance.The historical data methods [45], [46], [47] rely on analytics, but in this case the object of analysis is the previous task execution experiences. Though this information can be considered as subjective and relying only on a particular use-cases, still it is possible to extract the general task execution performance patterns. Despite to the historical data approach the online learning methods take into account only the recent task execution performance data and based on this knowledge orchestrate the computational processes using the specific resources, examples of such an approaches are provided in [48] and [49].
These methods of the workflow scheduling performance estimation also can be used in a certain combination, which gives us the hybrid class of the estimators. Example of such a system is the GraADS, which allows to create the computational models by the mixture of historical and analytical modelling methods. Another example of the hybrid estimator is provided in [50].
The Grid resources are prone to the failures but the workflow management systems are not dedicated to fix the failures, it is only possible to include in the workflow scheme the fault tolerance component. This component increases the probability of the successful accomplishment of the task execution even if part of the resources are failing.
According to the Hwang et al. [51], the failure handling workflow components can be classified into two main groups, the task and the workflow level failure handlers.
The task-level failure handling systems can be categorized into four subgroups (see 3.4).

The "Retry" approach [52] is based on a simply task resubmission approach. The task

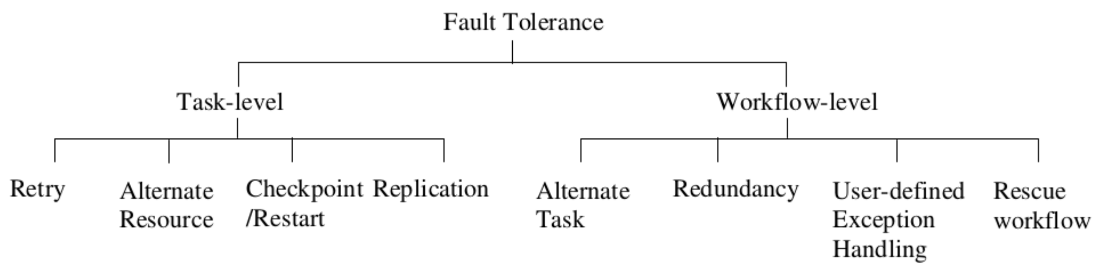


Figure 3.4.: Workflow Management System Fault Tolerance Classification [26].

might be submitted to the same or to the different resources as well. In case of the "Alternate Resource" method [52] the failed task, hence the workflow is resubmitted to the alternative resources. The "Checkpoint/Restart" approach [53] allows to shift the failed task on the alternative resources where it can continue its execution from the point of the failure. The "Replication" method [54], [51] can be cumbersome i terms of the resources consumption since it runs multiple copies of the same task on different resources and in this way decreases the failure risks, which might be caused by the infrastructure.
Among the workflow-level fault tolerance approaches there can be identified four subgroups (see 3.4) as well as in case of the task-level failure handling methods. For the methods belonging to the first three subclasses i.e. "Alternate Task", "Redundancy" and "User-Defined Exception Handling" of the failure handling workflow components, it is considered that there is a certain variety of the same task implementations in terms of the workflows.
The "Alternate Task" strategy is based on the sequential task resubmission with the available variety of the workflow parameters until its successful accomplishment or until the workflow variety is exhausted. The "Redundancy" method submits tasks with all possible workflow tuning parameters at once, which also can lead to the waste of the computing resources. The "User-Defined Exception Handling" approach makes possible for the Grid users to define conditions and the types of failures and corresponding rescue actions. The "Rescue Workflow" is oriented in identification of the faulty resources, which supposed to be excluded for following task execution workflows. This approach can be efficient in case of the large and reliable computing infrastructure where the share of the problematic resources is stably small.
The task execution steps defined in the workflow management system includes the communication with different types of the data storage and transfer systems. The systems providing data related services are independent from the computing resource management and other autonomous systems in most of the Grid environments. Hence for achievement of the high performance and throughput from the Grid systems it is essential to plan optimally the communication with data storage and transfer services in terms of the task execution workflow. According to [26] there are "User-directed" and

"Automatic" types of the data movement components in the workflow management systems. In "User-directed" case, the data movement to the computational resource and also the retrieval process of the results fully needs to be defined by the user, which requires good knowledge of the corresponding tools. In case of the "Automatic" methods there are three subclasses, the "Centralized", "Mediated" and the "Peer-to-Peer".
The "Centralized" data movement methods consist of one main component, which governs delivery of the data to the computing resource as well as it manages the resulting data retrieval process from it. The "Centralized" data management systems are good for the implementation in the small scale Grid environments with the short workflows.
In case of the "Mediated" data movement methods all data sets including initial, intermediate and the output of the task are registered and operated by the dedicated data management system. This approach is efficient for the large scale Grid systems with the large amount of the datasets.
In order to derive the final result of the data processing task, there might be required several intermediate steps of the computation, which also includes intermediate data processing. In case of the "Peer-to-Peer" data movement methods there is no need for the third party data management system. The data transfer among the suppliers and the providers is managed directly among the computing resources. This approach assumes that data management and transfer systems are integrated with the computing resource management software on each computational entity. Thus, the "Peer-to-Peer" approach can be efficient in some cases but it is related to the number of system deployment and the orchestrating difficulties.
Heterogeneous and distributed nature of the Grid infrastructure as well as the conceptual requirements of autonomy, scalability and QoS agreements makes difficult to manage the Grid systems in an efficient way. In order to avoid such difficulties there were developed several key components of the Grid for task execution process management. One of the components, the workflow management system, was already covered in a large extend. Another key component of the Grid systems in order to achieve the high efficiency and QoS in terms of the task execution and successful accomplishment is a Resource Management System (RMS) [55].
The main concern for the RMS is an efficient computational task execution. Despite to the Workflow Management System the RMS is rather oriented on a technical details of the task submission, monitoring and the result retrieval than on planning of the entire task execution process.
Due to the autonomy concern of the Grid sub-infrastructures i.e. the sites, it is important to organize the proper interoperability and the information exchange among the Grid and the site level RMS. The site level RMS is the computer cluster management system. The Grid site computational resources are located in the local network and do not have any access from the internet. Among the commercial and the open source local computing cluster resource management systems, there are several widely used systems like, the TORQUE Resource Manager [56], derived from the Portable Batch System (PBS) project [57], the Oracle Grid Engine [58], IBM LoadLeveler [59] and also the Microsoft Windows High Performance Computing (HPC) Server [60]. The Grid level RMS

communication with the cluster level systems allows to collect information to create the proper namespace of the available resources, also it allows to discover the available and accessible resources as well as makes possible to estimate the certain task execution duration and cost. All this information is important for the RMS in order to adjust its scheduling model and policy.
The RMS scheduling models can be classified into three groups the centralized, hierarchical and decentralized. This component of the RMS is very similar to the identical part of the Workflow Management System. The centralized scheduling model consists of one main part, which orchestrates the task execution on the available resources. The hierarchical model consists of several layers of the Resource Management Systems and the key aspect is interoperability among the layers. The decentralized model consists of several RMS systems without any subordination, the key aspect here is the interoperability and avoiding the resource conflicts among the resource managers.
The key point for the RMS scheduling policy definition is the possibility of dynamic change of the resource assignment to the particular task.
One of the first RMS was deployed in terms of the Globus Toolkit project, which served as a source for number of further developments of different Grid-related software standards. Nowadays the the Globus Toolkit 6 (GT6) [61] supports all required Grid components including RMS and makes possible to deploy the Grid infrastructure. Another pioneer software system example, which incorporated the RMS, is the Uniform Interface to Computing Resources (UNICORE) [62].

Data Management System

Different fields of science and the industry, relying on the Grid infrastructure demand not only for the computational resources, but also for the data storage and management services. Need for the data management systems is increasing especially for the following scientific fields: High Energy Physics, earthquake engineering, climate modelling, astronomy and medical research. In these directions the scale of the generated data has increased from tera-bytes to peta-bytes just in last several years. The expected scale of the amount of data for the next decade reaches exa-bytes.
Engineering, development and deployment of the data management systems for dealing with such an enormous amounts of information are related to the solution of certain number of challenges, such as:

- Data accessibility and visibility: The Grid resources i.e. distributed and heterogeneous infrastructure, is organized in different resource centers, so called sites. The Grid sites hold, manage and preserve the certain share of data. Site level data storage systems are not homogeneous and can be based on different approaches and solutions, but certain Grid protocols standardize the information retrieval functionality from the site storage systems in order to create the common Grid level namespace service. This service makes possible to manage the Grid level data preservation i.e. perform the data replication in different sites. The common Grid level namespace service also provides information about the locations of the dataset. The data replication service allows to decrease the risks of the data loss, which might be caused due to the hardware related problem at a particular Grid site.

- Transparency and the data access: The Grid level standardized data management protocols also allow to perform the multiple level authentication and authorization of the Grid user, without asking the credentials each time. This mechanism helps users to avoid repetitive identity check procedures in order to get access on a particular dataset which might be located in different sites, hence increases the data access and management efficiency.

- Efficient data transfer: Due to the large sizes of the datasets it is important to have an appropriate network infrastructure and systems to organize the data transfers to the computational resource, where the analysis of the provided dataset should be performed.

In the Grid the same data entity is stored in different sites with different, site-specific names. This fact causes number of problems first of all for the data discovery and secondly for the data consistency and sharing. The Grid level namespace service allows to create the data entity related meta-data, which assigns the single logical name to the data and also holds information about its physical locations. The namespace system also helps to organize the data replication functionality. The data replication i.e. copying the same data entity in different Grid sites helps to boost the data access process for its processing and reduce the network transfer overhead. Challenging for the data replication procedures is to keep the data consistency. Several copies of the same piece of information can be accessed in different physical locations independently, which also assumes the possibility of its changes. Therefore it is important to keep the track about all the modifications and hence replicate the changes in all physical locations and in such way keep the data consistency. The data namespace services as well as the replication mechanism and policies create the general feature of the data management system, the data accessibility and visibility. Examples of the systems with these functionalities are MetaData Catalog Service (MCS) [63], Replica Location Service (RLS) [64], GFarm [65], GridNFS [66], Don Quijote [67], etc.
In the transparent data access the key role plays the security aspects. Data access covers the aspects of the data transfers and the access management to the datasets i.e. only authorized users can manipulate the data. At the time of the initial Grid systems development there were two major data transfer protocols the FTP [68] and the HTTP [69], which were not sufficient in terms of the Grid Security Infrastructure (GSI) requirements. In terms of the Globus project the FTP protocol has been extended to the GridFTP [20] by means of incorporation the GSI concepts in it. For the completeness of the GridFTP functionality the dynamic client for the GridFTP, the UberFTP [70] has been developed. Another basic development for the secure data transfer among the Grid sites is the GSI-OpenSSH, which is the extension of the OpenSSH, similarly to the

GridFTP. Nowadays there are number of protocols which are already integrated in the Grid data transfer and management systems, e.g. Chirp [71], Data Link Client Access Protocol (DCAP) [72], DiskRouter [73] etc.
Besides the data transfer it is important to have an access to the data and to the storage space where the data are located. Taking into account the heterogeneity of the data storage infrastructure in terms of the hardware, software and the data access protection, it is not a trivial task to provide the unified interface to the users for all the data related manipulations. As a response to this challenge the Storage Resource Management (SRM) [21] was deployed. The SRM is not a file transfer service, it initiates communication with the corresponding data movement services in order to optimize the Grid site performance, avoids the overconsumption of the available storage space and guarantees the data storage space related manipulations e.g. space pre-emption, allocation and reservation. The SRM provides the unified namespace to access the dataset on a particular Grid site's storage resource and represents an additional layer between the Grid resource consumers and the complex, heterogeneous structure of each site.
Integration of the data transfer protocols and SRM with the GSI guarantees the transparent and the secure access to the data, distributed among the Grid sites. This can be considered as an approach for the solution of the data management related challenges in the distributed computing infrastructure.
Together with the security, data namespace organization and transparent access to the data it is important to have the efficient network transfer mechanisms to guarantee the in-time delivery of the data into the place where it suppose to be processed and analysed. Nowadays the network bandwidth and the connection speed among most of the Grid sites is sufficient for the data transfers among any the two sites, but still for some locations the network-related limitations remain. For such cases there can be used the overlay mechanism [74, 75]. The goal of the overlay mechanism is to optimize the data transfer and reduce the overhead required for in-time data provisioning to the location of the computational resources. Idea of this approach is to have the temporary storage spaces besides the computing centers, which would allow to optimize the data delivery among more than two computing resource locations. Examples of the overlay mechanism implementations are IBP project [74, 76], Kangaroo [77] and NeST [78].
Number of available open-source and commercial solutions for the common namespace organization in the heterogeneous storage infrastructure, possibility of integration the data transfer and accessibility with the GSI infrastructure, reliable and high speed network connection together with variety of overlay systems boosted the development of the Grid systems and network. Development and improvement of the data storage management systems remains one of the hot topics among scientific and commercial entities due to its key importance in digital era.

The Grid Information Discovery and Management System

The Grid infrastructure consists of number of computing centers, the number can be increased or reduced during different periods of time. Amount of available computational resources also varies, i.e. the number of available computing centers is not persistent in

time. Another reason of changes in terms of the resources is substitution of old resources with the new computing and storage capacities. Thus for efficient management of the Grid systems it is important to follow the dynamics of the available computing resource changes. Hence the Grid information and resource discovery systems plays a key role for the proper load distribution among available Grid computing sites that builds the basis for guaranteeing the proper QoS.
As most of the Grid related concepts and technologies, the fundamental and basic implementation of the Grid information system was performed in terms of the Globus Toolkit project. The Globus Metadata Directory Service (MDS) [79] and later MDS2 was one of the main conceptual implementations of such a system. The MDS2 held and was providing all the information about computational and storage resources available among a single VO. It has two core components: 1) The Grid Resource Information Service (GRIS) servers, implemented on the Grid site level and providing local information and 2) The main the VO level directory service provided by Grid Index Information Service (GIIS) servers.
The GRIS was providing the site specific information stored on LDAP [80] based directory system and on the site level there were no scalability or extensibility problems. For the completeness of the system functionality the site level information was collected by GIIS servers providing the VO level information. Though there were no performance related issues with the GRIS, once the amount of the site related information was increasing problems start occurring [81]. This problems were fixed with the further upgrades and modifications of the MDS system. The core modifications in the MDS system were performed in the MDS4 [82] version. This system has moved to the web-service based technologies, in particular it is based on the WS-Resource Framework (WSRF) and WS-Notification (WS-N) specifications [83].
The stable and efficient performance of the MDS was useful mostly for the Grid sites based on Globus Toolkit middleware, which also meant that the structure of published and aggregated information was not similar to the one that other Grid communities were using. This issue was causing interoperability problems especially among the Grid computing sites, which were deployed based on different middleware but were in the same VO. The solution for this problem was development of the common standard or standardized mechanism for the Grid information publication. First effort in this direction was initiated in terms of the GLUE (Grid Laboratory for a Uniform Environment) [22] project, which was a collaboration among European and US Grid projects and communities. Nowadays the initiative for the GLUE activities is taken by the Open Grid Forum (OGF) [84].
In terms of the GLUE project the concept of the information schema has been developed. The information schema is the standardized structure of the information for the Grid sites to publish the site-specific information. The schema allowed to organize the resource information exchange inside a particular VO even if the sites were deployed on the basis of different middleware. This was the big step forward in the interoperability improvement and this standard schema has been adopted by the number of Grid communities such as, LCG [85], OSG [86], NDGF [87], EGEE [88] and now by EGI [89].

The schema is an information format containing data about the resource endpoint names, their availability, current state, technical details and configuration parameters, as well as the list of authorized users and permitted actions. The GLUE schema was designed for both, the site and the VO levels, therefore the common data format is used on both levels of the Grid infrastructure, which increases the efficiency of the overall Grid resource management process.
The schema is structured by the number of classes and subclasses. This approach is sufficiently flexible, such that together with the required data fields one can include any specific detail about the particular Grid site or VO.
Two core classes the *UserDomain* and *AdminDomain* structures the schema and provides the hierarchically organized Grid site or/and VO level information. The remaining classes of the schema contain information about the service endpoints and related access policies, also about the computing and storage capacity. Flexibility of the information structure in terms of provisioning an additional Grid site or VO specific details is guaranteed by the optional data fields *Other Info* and *Extension Classes* [90].
All main and widely used Grid middleware implementations contain the Grid Information System as one of their core component. One of the oldest implementation is the MDS as a part of the GLOBUS Toolkit project, for the gLite the BDII [91]component serves as a site or VO level information provider as well as the endpoint for the resource capacity and state monitoring. In case of the UNICORE, the CIS [92] holds and provides the Grid infrastructure capability data. All these implementations are using common, GLUE schema as a structure for storing and publication of the information and mainly share the conceptual side of their functionality.
One of the widely used Grid information component in the LCG community is the Berkeley Database Information Index (BDII) 3.5. It consists of three main levels such as, the Grid site service level, the Grid site level and the VO level. The BDII component of the particular Grid site service level information provides details and data about the state, configuration and capacity regarding the particular service, this information is aggregated on the Grid site level BDII endpoint and published to the top-level BDII service. At all levels authorized users and services can query the information from the BDII components and end-points.

3.2.4. WLCG and ATLAS Distributed Computing

In this subsection the conceptual overview of the World-Wide LHC Computing Grid (WLCG) infrastructure and functionality will be provided as well as detailed description of the ATLAS Distributed Computing (ADC) systems functionality and interoperability will be covered.

WLCG

Initial structure and conceptual model of the WLCG was proposed in 1999 in terms of the MONARC [94] project. The basic idea of hierarchical and distributed organisation of the WLCG was so efficient and well-adjusted to the foreseen scientific challenges, that

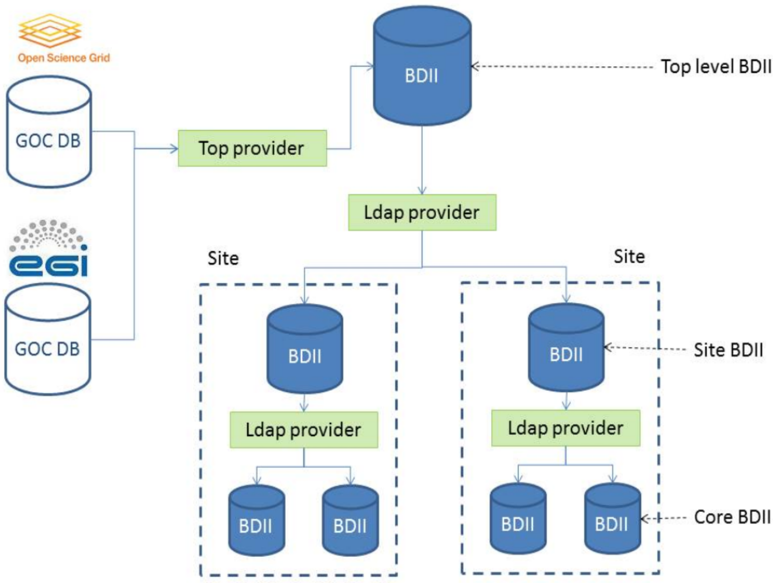


Figure 3.5.: Berkeley Database Information Index (BDII) [93].

it has not been changed much up to now.
WLCG represents cross-organizational Grid, it federates three large Grid communities, European EGI (the successor of EGEE), US OSG and NDGF, in total it is a collaboration of almost 160 computing centers of various capabilities from 39 countries. Main purpose of about 200 PB of storage capacity, nearly 350 000 computing cores and mainly 10GB/s network infrastructure of WLCG is to provide computational resources to the four main LHC experiments: ALICE [95], ATLAS [96], CMS [97] and LHCb [98].
The hierarchy of the WLCG is organized in so-called tier structure. The Tier 0 center resides at CERN [99] and its purpose is mainly to record an experimental data, perform the initial data reconstruction process and to distribute it to the computing centers belonging to the lower layer of the structure. Besides that, Tier 0 provides significant amount of the CPU cores for the raw experimental data calibration and also to support the data analysis process at the site.
The peak data rate produced during the Run 1 of LHC in 2012 reached ∼4 GB/sec. for ALICE, ∼1 GB/sec. in case of ATLAS and CMS each and ∼300 MB/sec. for the LHCb. This is expected to be doubled during the Run 2 starting in 2015, which in total is expected to reach 1 PB/s of data-flow from detectors directly. Of course such a big amount of raw experimental data is preprocessed and only after that saved at Tier 0 storage facilities, where the rate of the recorded data reaches 1.5PB/week. Due to the continuous flow of the experimental data during the LHC runs only 15% of the data remains at Tier 0 and the rest is distributed to the further levels of the tier structure and first of all in particular to the 11 Tier 1 centers. The purpose of the Tier 1 centers is to preserve and provide wide access to the experimental data for further analysis, reprocessing and distribution to the next computing and storage centers of the Tier structure

i.e. to the Tier 2 centers.
Provisioning of the storage and computing resources for the Physics simulation tasks and end-user analysis is the main functional goal of about 150 Tier 2 centers spread around the globe. The final layer of the Tier structure, the Tier 3 centers aim to provide mostly the computational resources rather than storage capacities see fig. 3.6.

The Tier structure wouldn't guarantee efficient functionality of WLCG without other

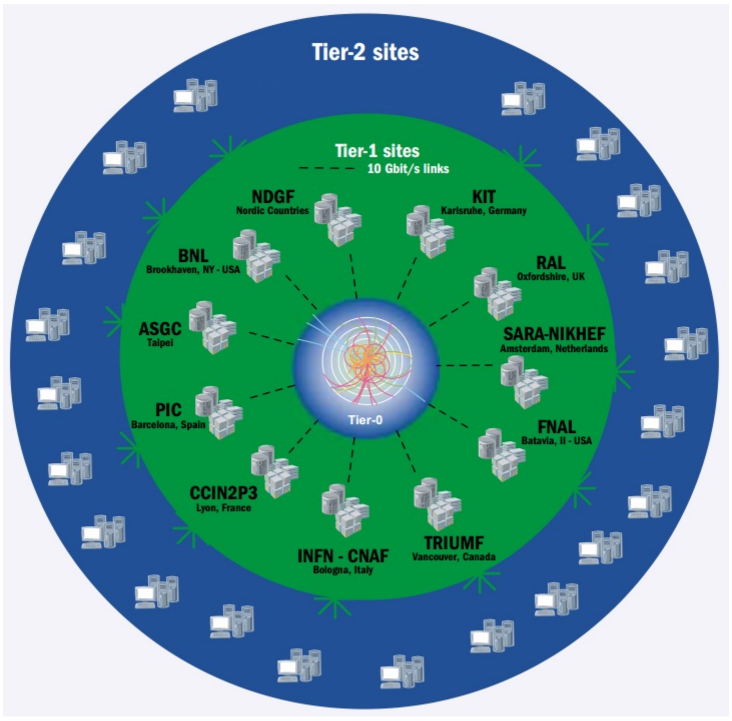


Figure 3.6.: WLCG Tier structure.

key components such as security and information systems, middleware and connectivity and operations infrastructure.
Security in WLCG is based on high level of trust among the Certificate Authorities (CA) i.e. VO units, which are agreed to issue and recognize the trust of the resource/host or user X.509 infrastructure based certificates. For efficient collaboration, vulnerability experience sharing and rapid reaction on security incidents an appropriate federated groups and teams are organized e.g. Grid Security Vulnerability Group, Operational Security Coordination Team etc.
The information system in WLCG plays key role in terms of interoperability inside the VO and also among the number of heterogeneous Grids. The structure of the information in WLCG is organized according to the GLUEschema standards. As for general WLCG site specifications, endpoint access Fully Qualified Domain Names (FQDNs) and available Grid services per site, as well as the information about the site downtime is available in the GOCDB [100].

Operations infrastructure of Grid consists of the Regional Operations Centers (ROCs). Each WLCG participant country operates one of such kind centers, but in some cases a single center covers several countries which belong to the same geographical location. The countrywide ROC centers help to improve operations among the sites on the country level. The common point for European ROC centers is the CIC portal [101]. Another spot supporting the communication of the Grid users and the site administrators is the GGUS [102], which is equipped with the web and mail interfaces. The GGUS allows the Grid users, operations managers and site administrators to report about the site related issues and follow the progress about each particular reported problem or task.
APEL (Accounting Processor for Event Logs) [103] is additional WLCG service, which supports the operations and allows to trace the number of submitted, successfully finished and failed computational jobs. Each WLCG site run an APEL instance, which publishes the job accounting information to the central server at RAL (Rutherford Appleton Laboratory) UK.
In order to follow the WLCG sites performance there were introduced two major metrics *reliability* and *availability* [104]. To observe the current site status and also check the history of these two major metrics the efficient monitoring infrastructure is required. For this purpose the SAM tests [104] and monitoring dashboard [105] systems were introduced among WLCG. These monitoring infrastructures perform tests for each of the services listed in the GOCDB under the particular WLCG site and transforms the measurements into the availability and reliability information.
The largest share of the WLCG sites operate with the EMI (gLite) middleware [106], the significant share of the resources also run globus and ARC [107] middlewares.
Stable 10GB/s interconnection among Tier 0 and Tier 1 centers as well as among Tier 1 and corresponding Tier 2 centers proved its efficiency already during the LHC Run 1. This stable and well-organized system allowed also to deploy number of improvements in terms of the WLCG operations automation and efficient management [108].

ATLAS Distributed Computing

The PanDA project as a Workload Management System (WMS) started in 2005 and up to 2008 it was mainly used for the ADC sites located in US. Due to its efficiency, robustness and security aspects also because of the plain python/https based REST framework providing plain client interface it became asan ADC-wide WMS since 2008 and soon after 2011 it started extension beyond the LHC experiments.
After becoming the ADC official WMS the PanDA system has been efficiently integrated with the OSG/EGI and NDGF middleware systems. Expansions of PanDA also forced provisioning of access to the cloud infrastructures like Amazon EC2 [109], Google Cloud Platform [110] as well as to other computing resource providers such as Oak Ridge Leadership Computing Facility (OLCF) [111], Future Grid [112], MIRA supercomputer [113]. Accepted computational jobs are distributed to the US OSG and EU EGI ADC sites using the Condor-G [114] system, which is integrated with PanDA as an "autopyfactory" module (see fig.3.7). In case of the NDGF sites the PanDA distributes jobs using another module, which integrates the ARC control tower functionality [115].

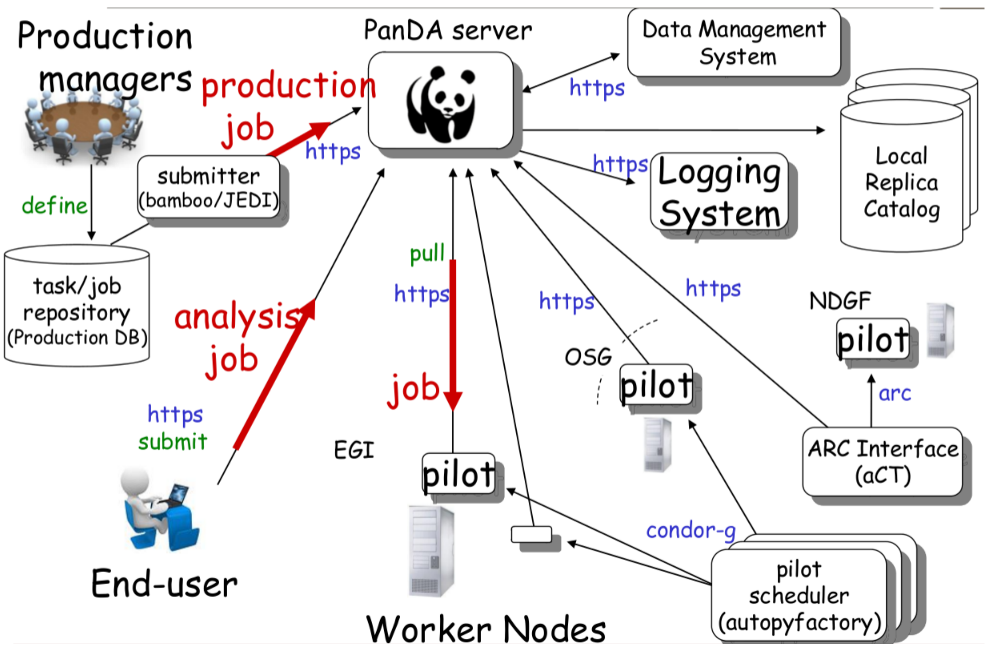


Figure 3.7.: The PanDA System Workflow.

It is important for resource saving and overall efficiency, that PanDA system does not send the code and the flow of computational task commands immediately after the computational job submission. At first it sends to the sites the so-called pilot jobs [116], which allocates certain slot of the required CPU resources and also checks the availability of the storage capacity. Once the resources are available the pilot job retrieves actual computational code from PanDA and launches it on the computing infrastructure. In parallel the job behaviour is monitored by the pilot and in case of facing the running time or resource limitation problems the job is cancelled and the sufficient information is reported in the system. The actual job status is dynamically changed and displayed in the monitoring web interface of the PanDA system and this information is again supplied by the corresponding pilot job, the job statuses defined in PanDA are "Defined", "Assigned", "Activated", "Starting", "Running", "Holding", "Transferring" and "Finished/failed", which are all self-explanatory and easily understandable for the owners of jobs. The pilot-based system does not guarantee the immediate execution of the job on the computing resource, it requests the resources but in order to avoid long waiting time the PanDA performs in advance assessment of the available computational sites by the brokerage module. This module analyses the data from the PanDA internal information system, which includes the number and priority of submitted pilot jobs per site, it also checks the amount of available CPU and storage capacities and information about the downtime of the particular site services. In case, the priority of the recently submitted job is higher it can skip the pilots with the lower priority at the site.Though pilots are used for all jobs submitted to the PanDA system, still there are differences among the

types of the jobs, which is reflected on the task queue organization per ADC site.
There are two main types of the computational jobs the production and user analysis. The production jobs perform the Monte Carlo simulations of the various physical processes and hence are used by the entire ATLAS community or by the particular working group. The user analysis jobs are mainly submitted by the single users who aim to process the particular set of the experimental data and extract the results. Despite to the analysis jobs the production jobs are submitted by the automated system governed by the production managers. Due to these specifications of the computational jobs in ATLAS experiment each Tier 1 and Tier 2 sites has at least one analysis and a production queue, which has an access to the site batch system via the Computing Element (CE) middleware [14].
Due to the deployment of high speed and stable network connections not only among the Tier 1 and subordinated Tier 2 centers but also among all computing facilities the ATLAS computing model has been modified from the hierarchical [117] to Meshgrid [118] structure. In the hierarchical model, due to the insufficient speed of all to all ADC sites interconnection the computational tasks containing certain number of computing jobs were submitted only within a single Tier 1 and corresponding Tier 2 infrastructure, which is called the national cloud. If the cloud's overall performance was high the risk of being overloaded was increasing while the certain well performing sites in other national clouds could have been idle. This is the problem of the resource optimal usage, solution to it became possible after the network connection improvements and introduction of the Meshgrid structure, which allows the direct transfers of the datasets among Tier 2 centers, which are not subordinated to the common Tier 1 center.
According to the available monitoring information the PanDA system is capable to complete 25 million jobs per month and even more, besides handling that bit computational resource management in 2013 via the jobs went through PanDA system 1.2 ExaBytes of data has been processed. These facts makes possible to mark PanDA as a first exascale system in High Energy Physics.
ADC occupies largest share of the WLCG for computing and for the storage capacity as well. Since the operation period of the ATLAS experiment i.e. ADC infrastructure it has proved its efficiency, which would be impossible without efficient functioning of the core systems, including the data management systems. The ATLAS Distributed Data Management (DDM) system is a standalone but well integrated infrastructure with the PanDA WMS and other core ADC services. The DDM stores/registers and provides access to the ATLAS experimental data. Based on its data replication and deletion functionality any kind of data loss as well as overloading of the distributed storage capacities with the identical data entities is not possible. The core component of the DDM system, which aims registering and cataloguing of the ATLAS data as well as its movement among ADC sites, deletion and consistency check of the data entities is called the Don Quijote 2 (DQ2) [119]. The DQ2 system proved its efficiency by its stable performance in terms of management of more than 150 PB of data, distributed in about 150 ADC sites. Though the DQ2 performance was sufficiently stable the operational costs of the system was increasing alongside with the growing volumes of the data requiring

efficient management, which was the hint to the fact that the system's performance limitation was reached. Therefore as a successor of the DQ2 the new system, Rucion [120] is to be commissioned.
However the core component of the DDM system is to be changed, the data entity structure and organization remains. The smallest component of the system containing the certain amount of an experimental data is a file, mostly consisting of C++ objects, suitable for reading programmatically. The files are organized in larger structures, called datasets. Datasets contain the fluent information about a particular colision in the ATLAS detector and its physical characteristics. The datasets itself are organized in the next level structure, the containers. The containers might carry information about the several collisions of the elementary particles which can be grouped according to some common features. The datasets and containers might overlap in terms of files. Frequently different datasets contain the similar files and when the datasets deletion is performed it is important to delete only the files, which are not in common with other datasets or containers. Therefore deletion process can not be considered as a trivial process and certain dedicated policies are defined for it in DDM system.
There are several components of the Rucio system that represent its core functionality and the concept. The Rucio system performs operations using the accounts, each user of the system, the human, automated Monte-Carlo production robot or a working group is mapped to the particular account or group of accounts. Association of the system users to the accounts is performed based on X.509 certificates or using the Kerberos token. After the authentication Rucio performs the authorization in order to define and verify the permitted actions for the particular account.
The data entity naming schema in Rucio system has the conceptual meaning, each of the dataset or file has two core components of the name, the scope i.e. identifier for the realm of activity where the data will be used e.g. production or user analysis and the second part the associated name in the system. Each of the data entities has a characterizing attributes. In case of file attributes, there might be provided an information about their availability, while for the datasets it might be given a hint if it is possible to add the content i.e. files to it or not. Also there are metadata attributes defined by the users (number of events, run number, run period or job id) and by the system (size, checksum, creation time, etc.) all these attributes are required for the efficient usage and Rucio performance improvement.
The single unit of the Rucio distributed storage capacity is called the Rucio Storage Element (RSE), which has its own attributes representing available data access protocols (SRM, HTTPS), storage type (disk or/and tape), information about the free, used and non-pledged storage capacities, weight value, which is used for the data distribution and deletion operations optimization.
Each file stored on a particular RSE can be identified by its Physical File Name (PFN). In order to reach the data stored into the file from outside i.e. from other RSE or by the user it is important to have the corresponding identifier, which can be the file name, URI or any other identifier adopted in the system. The mapping between the data entity and the PFN is implemented in RSE as a separate functionality.

The data replication mechanisms in Rucio is implemented based on the replication rules. Each rule is owned by the account and defines the number of the data replicas that need to be available on a certain set of RSEs. The replication mechanism also assumes that the files need to be organized in a certain hierarchy in order to optimize the data analysis code execution, which is I/O intensive. Data deletion and subscription mechanisms are also tightly connected to the accounts and rules concept [120] as well as to the hierarchical organization of the data.
From the architectural point of view Rucio consists of four core components the server, users interface, resources and the daemons [120].
The server component of the Rucio infrastructure supports communication of the several core constituents of system and provides the REST interface to the users. Also it contains the user authentication and authorization mechanism that maps users to the accounts based on Oauth [121] approach.
Users can communicate with the server via the command line client that represents the user interface of the system. Besides the communication the clients provide the Python based programmable interface that simplifies the integration of the users specialized requests into the code and also into the DDM system.
As for the resources component of the Rucio architecture one can refer to the RSE abstraction layer, which hides the complex and heterogeneous distributed software and hardware storage infrastructure. The abstraction layer allows to interact with the any kind of RSE with random location using the standardized tools and applications, which simplifies users interaction with the data.
There are five key daemons of the Rucio that puts together number of modules and components of the system together. The daemons are *Conveyor* - supports the file transfer operations, *Reaper* - daemon for the replica deletion, *Undertaker* - provides mechanism for the expired data identification, *Transmogrifier* - monitors and applies replication rules to the newly created datasets, *Judge* - provides functionality for creation of the data replication rules.
As an important component of the system the SQLAlchemy (need the reference) can be considered, since it allows flexible functionality of the Rucio with the widely used RDBMS e.g. Oracle, MySQL or PostgreSQL.
All these Rucio components and features as well as its well integrated infrastructure with other core ADC services guarantees and already demonstrates its stable performance since the beginning of 2015 i.e. since its commissioning time.
Technical and general information about ADC resources is dynamic and changes in time, therefore it needs to be updated periodically in core ADC systems. This information plays key role in efficient management of the entire infrastructure. The set of technical details, being part of the site related information, is important for proper functioning of the PanDA and DDM central services. The ADC sites are part of different Grids such as OSG, EGI and NorduGrid and hence information about the sites is distributed in different information systems and doesn't share the common structure, which was the main motivation for the development and deployment of the information system called AGIS (ATLAS Grid Information System) [122].

AGIS provides information about ADC topology, sites availability, also FQDNs of the site service back-ends, the site downtime information, site's parameters in the PanDA system, datasets replication details, users privileges and details about their role and authorized actions for the associated accounts. All this information is stored and managed by the Oracle Database Management System. AGIS provides ways not only to view the aggregated and stored information but also to reuse it via the Application Programming Interface (API), which is the way how the system is integrated with other central ADC services. The Django framework [123] based WebUI is the way how the information can be viewed and changed. The changes are not reflected in the source information systems, but since AGIS represents the basic entry point of information for the ADC central services, changes at this level are sufficient for proper functioning of the entire infrastructure. Command line tool is yet another way for information retrieval from AGIS.
Aggregation of the ADC site specific key information in a single point, mapping of different technical and general site related attributes in terms of a single object, multiple ways and formats for accessing and reusing of the collected and organized information, simple WebUI based approach for changing and viewing of various details about an ADC infrastructure, turns AGIS not only in a core service for ATLAS computing but also it became a important component for automation of the ADC operations.
Dedication of the WLCG and hence the ADC infrastructure is to store, preserve and analyse extremely valuable physics experimental data. Distributed infrastructure of ADC is very complex and heterogeneous, it is prone to hardware and/or software failures that might cause the data loss, violation of the QoS agreement and general efficiency degradation, which is not acceptable. In order to avoid the large scale problems among the ADC systems and core services, the constant, 24/7, monitoring and reporting is required, which is the main responsibility of the ATLAS Distributed Computing Operations Shifters (ADCoS). Shifters are mainly personnel affiliated with various scientific institutions and computing centers who are involved in the ADC. Their main duties are related to the system deployment and administration activities, at their sites. Besides the shifters there are ADC central services and operations responsible who are informed about critical issues in order to take an appropriate actions. The action taking responsibilities are distributed according to the authorisation rights, therefore in some critical situations also the ADCoS shifters, with different roles are able to take particular actions.
The ADCoS shifts are organized in three time zones, Asia-Pacific (00:00-08:00 hrs CET), the Europe (08:00-16:00, CET) and the US (16:00-24:00, CET). During each time zone shift, there are always several people on line, who mainly communicate with the dedicated chat program and take joint decisions about the various issues. Advantage of the dedicated chat program is to store the communication history, such that the next time zone shifter can follow the details about the issues or statuses of the entire ADC infrastructure and services. Each time zone shift has a Shift Captain, who coordinates the communication among the Senior, Trainee and Expert shifters as well as with responsible for the ADC central services management. There is one time slot per day for

the Expert shifters, who have the sufficient experience and knowledge about the current activities, issues and the structure organization of an ADC systems and services. The Expert shifters serve as a connecting point between the time zone shifters in order to transfer the details about the ongoing issues which needs and action taking or further observation and reporting. There are three slots for the Senior shifter, each per time zone. The Senior shifters should actively check the current status of the ADC site services, downtime information, data distribution and task execution as well as the status of the central services such as the central data base systems located at Tier 0, Castor storage system, Squid instances for the CVMFS and various systems and services supporting the DDM and PanDA functionality.
Since there is a certain level of personnel migration from and to the ATLAS community it is important to have adequately trained and experienced human resources for supporting the ADCoS shifts, therefore the special Trainee shift slot exists, each per time zone. The Trainee shifters have to be active and need to follow the issues and also learn the tools and rules of the ADC operations mainly from the Senior and Expert shifters. Activity and the learning rate of each of the Trainee shifter is evaluated by the Senior shifter who are taking the shifts together with the Trainee. The evaluation is reported to the ADC coordinators in order to take a decision of promotion of a persons from a Trainee to the Senior status. The training period usually requires from 10 to 12 sequential shifts, without interruption.
The general tools for the shifters in terms of problem identification among the ADC central services or among the infrastructure is the set of monitoring systems, which needs to be checked and analysed constantly. There are two types of monitoring information providing the real-time monitoring data and also the tools aggregating the long-term performance details of the ADC sites and services. The real-time monitoring data shows the current state and the status of data transfers from and to the sites, also the performance of the CPU farms, as well as the details about the computational tasks that are registered in the WMS system with certain status and in addition the CERN Analysis Facility performance and the status is also presented. Information about the central services displayed in the SLS (Service Level Status) monitors is also the part of the real-time monitoring information. The long-term performance details are stored for further analysis and bottleneck detection in the ADC systems and services. In general, the tools providing such kind of information store the summary data from the real-time monitoring information and provides it in different known formats e.g. XML or JSON (JavaScript Object Notation) for their further analysis using Apache Hadoop or Apache Casandra tools as well as using other machine learning or statistical methods and applications.
Actions taken by the shifters are mainly based on the real-time monitoring information and it includes ADC site exclusion from the computational or data transfer activities. Besides manual interaction there are certain number of automation tools for the site exclusion from the activities in case of its downtime or its poor performance. The site performance issues in terms of PanDA system are detected using the HammerCloud [124] tests. The HammerCloud system submits the trivial and pre-checked scripts i.e. test

jobs to all ADC sites providing the computational resources. The jobs are analysing the facility with number of tests and in case of the certain level of failures the site is temporary excluded from the PanDA system activities and the corresponding site responsible are notified. Once the issues are solved at the site the HammerCloud system automatically activates the facility again. The site exclusion by the HammerCloud or by the shifter is reflected on the site availability and reliability report, which shows the optimal usage of computational and storage resources for the ATLAS data analysis and preservation. The site exclusion from data processing or storage activities by the shifter is performed based on the analysis of the computational job related log files and messages available in the PanDA system. In case of the certain signature in the log files or the failure messages there can be identified the site related issues, which can serve as a basis for the site deactivation in the PanDA by the authorized shifter. Another core source showing detailed information about the data transfers efficiency from and to the site as well as the data deletion efficiency is the DDM dashboard [125]. Information provided by the dashboard also can be the basis for the site manual exclusion from the data transfer related activities. The information contains the reason of the data transfer or deletion failures and in case of the certain key words it is possible to identify whether the failures were related to the site infrastructure problems or it was caused by the central services malfunctioning.
The ADC monitoring tools as well as the ADCoS play important role for several years in providing the reliable and stable services for the ADC infrastructure consumers. Though, the human interaction and involvement in the ADC operations is important the goal is to reduce the human resources and factor for such activities and automate the infrastructure management processes as much as possible. The automation should be performed on the basis of the existing information provided from the monitoring tools as well as on the basis of the accumulated experience by the shifters.
Efficiency and reliability of the Grid computing concepts has been proved to be a consistent with the requirement and challenges mankind is facing now. The best proof of the ideas reliability is the WLCG, which represents the largest Grid infrastructure in the world. Besides the good overall performance of the WLCG systems and services, important role in the proof of the concept has played its particular components such as ATLAS Distributed Computing.
Alongside with the building and evolution of the Grid concepts, it helped to emerge number of new research activities in the field of computer science. Some of them ended up with the strong new concepts such as Cloud Computing and some of them continues evolution like the Autonomic Computing concept. The variety of the successful Grid systems also proves the importance and strength as well as the correctness of the Grid computing core ideas.

4 ATLAS Tier 2 Center GoeGrid

It is impossible to imagine modern scientific and research activities without computing. The scientific and research groups at the Georg-August-Universität Göttingen are not an exception, hence the joint initiative to deploy the common computing and data storage facility arose, which was realized as a computing and data storage centre GoeGrid [126]. The main contributors to the joint computing centre up to now are faculty of Physics and in particular II. Physics Institute [127] and Institute of Theoretical Physics [128], MediGRID [129], TextGrid [130] and *Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen* GWDG [131], the computational resource distribution among these communities is provided in the table 4.1.

Besides serving as a common computing and storage resource for the scientific groups

Community	HEPSpec2006	Fraction(%)
II. Physics Institute	17092	66.8%
Institute for Theoretical Physics	5300	20.7%
MediGRID	2564	10.0%
TextGrid	448	1.8%
GWDG	171	0.7%

Table 4.1.: Apportionment of the GoeGrid computing resources by the contributors.

at the Göttingen University, 66.8% of the facility i.e. the share for the II. Physics Institute has the status of WLCG Tier 2 centre since 2008. The detailed description of the hardware and software setup of the Tier 2 centre, GoeGrid, as well as description of supplied services, monitoring and management tools will be provided in this chapter.

4.1. Hardware Setup

The computing facilities involved in WLCG at different Tier level provide mainly data storage and computational services. The computing centres at the Tier 2 level also provide information services for the Grid information system as well as accounting information about the resource consumption and monitoring data. These services are provided via the adopted and standardized middleware components, while decision for the computing facility management and monitoring are taken individually and from this point of view each site differs from other.
The computational service provisioning is guaranteed by the batch system, consisting of the head node, managing the computational tasks distribution among the worker nodes, which execute the actual data analyses code and return the results. The GoeGrid consist of 317 heterogeneous worker nodes, with 634 CPU-s and 3496 cores (see table 4.2) with at least 3-4 GB of memory per core. All the compute nodes are interconnected via the 1GB/s local network, without link to the internet. Each compute node is equipped with two identical hard drives integrated in RAID 1 disk and used for to run the operating system. In case of the entire batch system the CentOS 6.6 x86_64 [132] is the default operating system.

For the WLCG Tier 2 computing centre the storage space is managed by the dCache

CPU	**Clock[GHz]**	**# of CPUs**	**# of Cores**
Intel Xeon E5-26650	2.40	32	256
Intel Xeon E5530	2.40	32	136
Intel Xeon X5355	2.66	152	608
Intel Xeon X5650	2.67	104	624
Intel Xeon E5-2660v3	2.60	12	120
Intel Xeon E5-26600	2.20	32	256
Intel Xeon X5650	2.66	8	48
Intel Xeon E5-2650v3	2.30	30	300
Intel Xeon E5-2660v2	2.20	26	260
Intel Xeon E5-26700	2.60	16	128
Intel Xeon E5440	2.83	190	760

Table 4.2.: GoeGrid CPU specification.

system [133], where the data from the ATLAS experiment [134] and related simulation tasks are stored and analysed. The dCache system at the GoeGrid aggregates the storage space provided from the 12 independent servers as a single capacity of 1.1PB size, which is divided into the corresponding space tokens managed by the responsible personnel for the ATLAS experiment data management system. Besides the dCache, around 25TB of the Storage Area Network (SAN) is also in use mainly for hosting the images of the virtual machines and also for supporting the Network File System (NFS) mounted on each of the compute node of the batch system as well as at the storage servers. The servers constituting the storage of the Tier 2 centre GoeGrid, have one 1GB/s network interface accessible to the global internet and 10GB/s network card for the local area network, where the most communication from the worker nodes comes to.

4.2. Software Setup

In order to provide the requested computational and storage services as WCLG Tier 2 centre GoeGrid the standard and adopted middleware packages are in use.For the hardware and software level management as well as for the monitoring of the entire infrastructure, different software packages and suits are deployed. Description of the software packages for the WLCG service provisioning as well as for the monitoring and management will be provided in this section.

4.2.1. The Batch System

The batch system of the GoeGrid is deployed based on the TORQUE [135] resource manager and MAUI [136] computing job scheduler packages. The TORQUE resource manager is a client-server application and consists of two main parts the *pbs_server* daemon running on the head node of the batch system and the *pbs_mom* daemon running on each of the multiple compute nodes. The *pbs_server* daemon at the GoeGrid is running on a dedicated machine alongside with the scheduler MAUI. The *pbs_server* accepts the request for the resources required for a particular computational job, which is transmitted to the scheduler. The scheduler, according to the predefined scheduling strategy, returns the list of the available worker nodes with the desired resources to the head node daemon. Based on the provided hosts list the tasks are submitted sequentially from top to the bottom. Overall efficiency of the batch system is guaranteed by the constant communication among the head and worker nodes, the *pbs_mom* daemons supply the head node and hence the scheduler with the information about their availability and about the status of the submitted job. Another key aspect affecting the batch system functionality is the scheduling algorithm. Certain amount of the scheduling strategies are integrated in MAUI. In case of GoeGrid, due to the sharing of the resources by multiple scientific groups, the fairshare scheduling algorithm [137] is used. The fareshare algorithm guarantees availability of predefined amount of resources, but in case more computational power will be required and meanwhile there will be idle worker nodes, jobs will be submitted there as well. In this way the most optimal resource consumption can be reached for all scientific groups.

4.2.2. Storage System

The data storage infrastructure for the GoeGrid Tier 2 centre is based on dCache system. The dCache software is the results of joint effort of the *Deutsches Elektron-Synchrotron (DESY)* [138], Hamburg, *Fermi National Accelerator Laboratory (Fermi-Lab)* [139] Batavia, Illinois, Copenhagen and Linköping Universities.

The dCache system allows to create a single storage space with a common file system from independent servers interconnected with the network. The dCache file system *purely normal file system (pnfs)* [140] allows to access and modify each data object without knowing of its physical location on the storage servers. Due to the ATLAS Distributed Computing (ADC) organization it is required to have the centralized access to the certain parts of the entire disk space of Tier 2 GoeGrid. In case of ADC all these functionalities and actions are possible via the SRM protocol, which is guaranteed by the dCache system. For the data transfers dCache provides set of standard and adopted protocols for the Grid sites, such as: dCap, GridFTP, GSIdCap, HTTP (WebDAV) [141] and XRootD [142].
The building block of the dCache functional structure is the cell service, running a dedicated Java Virtual Machine (JVM). For each particular functionality e.g. providing the data transfer using a particular protocol, storing a data object etc. is handled by the dedicated cell services. Organizing of the dCache structure based on the cell services is due to its Service Oriented Architecture (SOA) [143], hence the cell services are able to communicate with each other over the network infrastructure. The list of the dCache key services is provided in the table 4.3.

Component	**Functionality**
Data Input/Output, I/O door	Access to the data objects on the storage system; Single door for management and protocol: administration, dCap, GridFTP, GSIdCap, HTTP, SRM, Web-DAV and XRootD.
pnfs manager	Provides the common name space for all data objects stored in a single dCache system. Crucial part in the dCache system, it's stability guarantees the entire dCache system serviceability.
pool	Actual storage for data
pool manager	Provides actual file transfer functionality, interacting with one or several pools of data, identifying pools, provides pool ranking using the cost metric
gPLAZMA	Grid-aware PLuggable AuthoriZation Management Security mechanism for user authorization over: kpwd, Grid-mapfile, gPLAZMAlite-VORole-Mapping, Sams-VO-mapping

Table 4.3.: Key cell services in the dCache system.

One of the key functional services, which does not have a corresponding cell service is a pnfs manager (*pnfs*). The *pnfs* has a complex structure and it guarantees the dCache

functionality by means of provisioning the common file system. Each data object stored in the dCache system is registered in the *pnfs* with its unique, 96 bits ID. All the IDs are stored in the dedicated *pnfs* database suite, which has corresponding flexibility to handle the large amounts of data objects (up to 2^{77} entries).
Among all available data transfer protocols supported by the dCache system the key importance for the ADC users have the dCap, GridFTP and the XRootD. For dCap and GridFTP protocols authentication is performed using the GSI infrastructure integrated with the dCache system, while XRootD represents the completely unauthenticated protocol used for read-only access on data. In case of ADC the dCap protocol is mostly used for the data transfers among the worker nodes and storage system i.e. at the LAN level, while GridFTP being part of Globus toolkit supports the parallel transfers of a single data object and hence is more widely used.

4.2.3. Middleware Componenets of GoeGrid

The Computing Element (CE) is one of the core components of any computing facility involved in the Grid computing. In case of ATLAS Tier 2 centre GoeGrid the middleware used for CE role is *Computing Resource Execution and Management* (CREAM) [144]. For the fault tolerance there are deployed two instances of CE running the CREAM middleware, which is a Java based application running in the Tomcat application server environment and have the Service Oriented Architecture.
In ADC environment the role of the central WMS has the PanDA system, which distributes the submitted computational tasks to the computing facilities CEs. Communication between the PanDA WMS and CREAM is possible over the *Interface to Cream Environment (ICE)*, which is the specific, integrated component of the middleware. Besides the WMS, computational jobs can be submitted to the CREAM directly by the user, hence job submission tools and applications like Ganga [145], pAthena [146] and prun [147], adopted among the ADC users, are supported by the CREAM. Besides the job submission the CREAM, via its interfaces, provides the control mechanism to monitor, cancel, abort, resubmit or suspend the job. Besides the job related functionality the manipulations for the users with the managerial access rights are guaranteed, hence the user with the granted permission can enable or disable the job submission on a particular CREAM instance also collect the internal, service related, information. The key aspect for the CREAM is the communication with the corresponding computational farm, which is possible via the *Batch-system Local ASCII Helper (BLAH)* [148] component. The BLAH component, integrated in the CREAM guarantees the communication with the LSF, PBS/Torque, Condor, SGE and BQS batch systems (*Local Resource Management Systems (LRMS)*). The computational jobs submitted to the CREAM are written using the classad-based Job Description Language (JDL) expression [149]. The JDL allows to describe the jobs requiring the batch and parallel processing i.e. there might be requested a single or multiple CPU cores for a single job. The CREAM supports both types of jobs which are transferred to the LRMS directly, there is no specific requirement for the batch jobs in terms of the software while for the parallel jobs, the MPI library is important, that allows the interprocess communication.

Management of computational resources plays important role in an efficient resource management. In case of WLCG as well as for ADC information about the provided and consumed CPU resources is known for each computational site. In case of GoeGrid as well as for other ADC sites the CPU accounting information is provisioning is organized via the APEL service. The accounting information contains the number of submitted and successfully finished jobs at the site as well as the associated core consumption information. All the information is aggregated at the central server at Rutherford Appleton Laboratory (RAL) [150] and it is possible to filter these data by user, VO, region, country etc. Information aggregation at the central server is organized according to the push model, which assumes that each computing facility contributing to the ADC runs its own, local instance of APEL server that pushes the accounting data to RAL.
The locally deployed APEL server contains two major parts the APEL parser and the APEL published. The APEL parser communicates with the site computing elements in order to get the access to the batch system log files, which contain information submitted jobs and their states. This information is analysed in conjunction with the benchmark values provided from the CEs and are recorded in the local database. The APEL publisher access the analysed accounting data in the database and pushes the data to the central APEL server in order to contribute in overall WLCG CPU resources accounting report preparation. The local instance of APEL server at GoeGrid is running on the KVM virtualization environment [151] with a single virtual CPU and 1GB of memory.
The Tier 2 centre GoeGrid provides multiple Grid services with corresponding FQDN and ports, also the information about the number and configuration of CPU cores and memory per core is changing from time to time. Provisioning of detailed site related information and specification in a centralized way the BDII service has been deployed. The BDII instance is running at GoeGrid on a dedicated virtual machine, deployed in the KVM environment similarly to the APEL instance. The BDII service of GoeGrid provides information about the number of available cores at the site, the FQDN and port information for storage related and other grid services required by the ADC management.
In order to support the proper Grid level authorization and authentication as well as the functionality of all tools needed adopted in ADC for the data analysis the worker nodes and the user interface middlewares are also deployed on the worker nodes of the GoeGrid batch system.

4.2.4. Management Tools

Provisioning of highly reliable and available services by the Tier 2 centre GoeGrid depends on efficient management of the hardware, especially the most consumed part, the worker nodes of the computing farm. Frequently it is required to reinstall a single or several worker nodes or upgrade all of them, which is not feasible manually, when dealing with more than 300 machines. The Rocks Tools [152] at GoeGrid, provides first of all the automatic installation mechanism including deployment of all needed software packages. The XML based tree structure of Rocks Kickstart Graph [153] is employed to define the RedHat Kickstart file [154], which allows higher flexibility in automated OS and

software suite installation or upgrade. Besides the efficient Kickstart file deployment, Rocks Tools at GoeGrid holds the information catalogue with all the key specifications of the deployed servers. Via the dedicated CLI interface of the Rocks Tools, one can get the GoeGrid hardware details immediately, which is very useful for the resource accounting and bookkeeping. In addition to all the mentioned important features, Rocks Tools provides the automatic DNS server configuration functionality. In case of the GoeGrid this feature is used for the LAN environment, which is important especially in case of the computing farm. Once the details about the new host are entered in the Rocks database the new DNS entry is generated and it becomes resolvable without any additional manipulations on the DNS server configuration file. Besides the functional features, Rocks also enables Ganglia [155] monitoring tool integration in the hardware under the management. Once the new worker node is deployed or the OS is installed or upgraded, it is automatically included in the gangalia monitoring, that allows system administrators to follow the load of the machine in a real time. All these features of Rocks Tools makes the management of available resources very flexible and efficient.
Another tool that is actively used in the GoeGrid for its administration tasks automation is the CFengine [156]. The CFengine is a distributed system, widely used for the monitoring, analysis and management of large scale computing facilities. The main motivation of the CFengine deployment in GoeGrid environment was the automation of big number of routine administrative tasks requiring large share of manpower. CFengine allows to translate most of the tasks as set of rules, which are called policies. The policies can be general and specific for each instance, including storage system and computing farm servers as well as for other grid service provider components. All these policies together build the model providing the required state and configuration for the key services for general infrastructure and also for a particular server. The set of policies are written in the text file in form of the rules using the CFengine specific language. Each of these text files are distributed on a dedicated machine and the CFengine service checks the different parameters and/or states of the particular components and services according to the predefined set of policies. The set of policies for entire infrastructure and for a particular server represents the model of the entire system behaviour, which is important to control for the system administrators. The CFengine also allows the detection of the divergence from the described model, which helps in time problem detection and isolation, hence increases the overall performance.
The Rocks Tools and CFengine are important applications that simplify the overall management and administration process of GoeGrid. Both of these tools guarantee the stable performance and functioning of GoeGrid for more then six years.

4.3. Monitoring

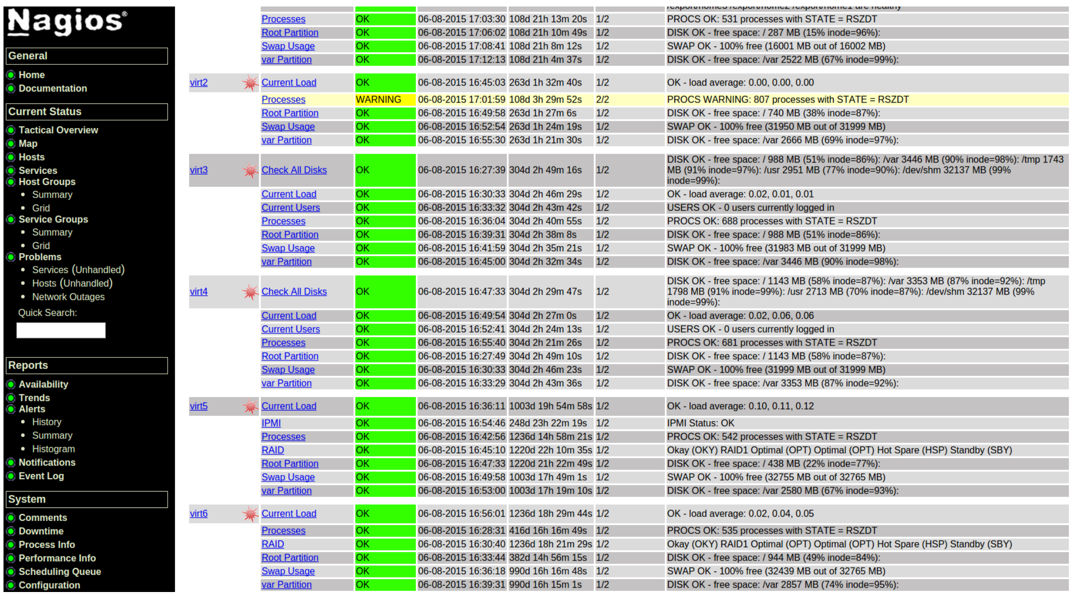
General description of the monitoring tools for the site, general description of the concept of the large computing facility monitoring. Taxonomy of the monitoring tools. It is impossible to perform efficient management of any computing facility without the corresponding monitoring tools and applications. Monitoring is required to keep an eye on hardware components, services and also load of entire infrastructure. In this section description of monitoring tools used at GoeGrid will be provided in more details.

4.3.1. Nagios

The Nagios software [157] is an open source network, systems and services monitoring software. It allows to check network availability, also if a certain service port is functional and reachable, it also shows hardware component information such as CPU load, hard drive status, RAID status, usage of certain partitions on disk etc. The Nagios has a plugin structure, each plugin executes a certain monitoring task i.e. checks the service availability or a particular hardware component functionality. In case of GoeGrid 29 servers are under the Nagios monitoring. Most of the servers run the dCache storage management system, the rest of them are dedicated for providing the nfs service to GoeGrid infrastructure, as well as supporting the KVM virtualization, in addition the Apel and BDII instances are included in the list of the monitored hosts.
In case of GoeGrid all the plugins distributed on 29 servers communicate with a Nagios server using the *Nagios Remote Plugin Executor* (NRPE) [158]. Information pushed to the head server is analysed and four categories and corresponding colour codes are used to identify the problematic case with the key work CRITICAL and a red color, dangerous but not CRITICAL state with the key word WARNING and a yellow color, the state when the service or hardware under the monitoring is in acceptable state the OK key work is used in conjunction with the green color (see fig. 4.1). The fourth state corresponds to the situation when the host is not reachable or the response from the NRPE client is not interpretable, for such cases the Nagios uses the key word UNKNOWN and a gray color. Also there might be CRITICAL states when the entire host or a particular plugin is not reachable or malfunctioning.
The web interface of Nagios displays the status and the corresponding message of each particular plugin. The results of plugins are grouped under the corresponding host, that simplifies a problem detection for the computing facility administrators. The left side menu allows to display the current status of entire computing facility or a particular server in different ways, the further menu options also allow to group the problematic events by hosts and other characteristics. In addition it is possible to generate the overall performance reports for a particular host or group of hosts.
The Nagios monitoring tool is online monitoring system providing the current state of the entire infrastructure in a very efficient way. Its alarm and notification system also makes Nagios the key component of the GoeGrid management process.

4.3.2. Ganglia

The GoeGrid computing farm i.e. the batch system server and the storage system components with APEL and BDII instances are under the monitoring of two separate Ganglia monitoring servers. The Ganglia instance running for the GoeGrid batch system is automatically configured and deployed in terms of Rocks Cluster application, while the hosts which are under the Nagios monitoring are included in the Ganglia by the GoeGrid



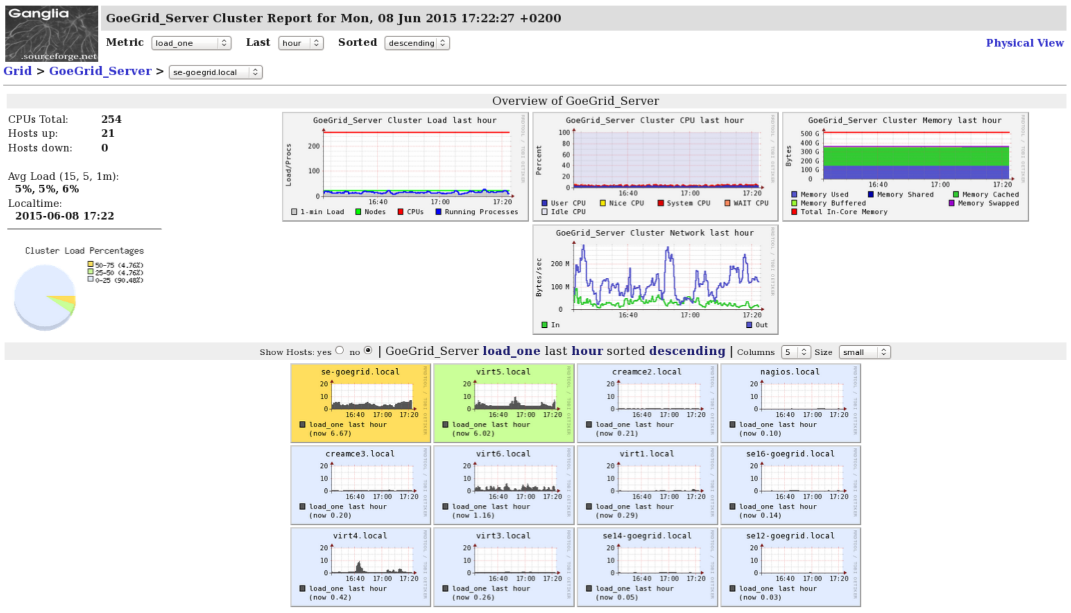
						/export/home3 /export/home2 /export/home1 are healthy
	Processes	OK	06-08-2015 17:03:30	108d 21h 13m 20s	1/2	PROCS OK: 531 processes with STATE = RSZDT
	Root Partition	OK	06-08-2015 17:06:02	108d 21h 10m 49s	1/2	DISK OK - free space: / 287 MB (15% inode=96%):
	Swap Usage	OK	06-08-2015 17:08:41	108d 21h 8m 12s	1/2	SWAP OK - 100% free (16001 MB out of 16002 MB)
	var Partition	OK	06-08-2015 17:12:13	108d 21h 4m 37s	1/2	DISK OK - free space: /var 2522 MB (67% inode=99%):
virt2	Current Load	OK	06-08-2015 16:45:03	263d 1h 32m 40s	1/2	OK - load average: 0.00, 0.00, 0.00
	Processes	WARNING	06-08-2015 17:01:59	108d 3h 29m 52s	2/2	PROCS WARNING: 807 processes with STATE = RSZDT
	Root Partition	OK	06-08-2015 16:49:58	263d 1h 27m 6s	1/2	DISK OK - free space: / 740 MB (38% inode=87%):
	Swap Usage	OK	06-08-2015 16:52:54	263d 1h 24m 19s	1/2	SWAP OK - 100% free (31950 MB out of 31999 MB)
	var Partition	OK	06-08-2015 16:55:30	263d 1h 21m 30s	1/2	DISK OK - free space: /var 2666 MB (69% inode=97%):
virt3	Check All Disks	OK	06-08-2015 16:27:39	304d 2h 49m 16s	1/2	DISK OK - free space: / 988 MB (51% inode=86%): /var 3446 MB (90% inode=98%): /tmp 1743 MB (91% inode=97%): /usr 2951 MB (77% inode=90%): /dev/shm 32137 MB (99% inode=99%):
	Current Load	OK	06-08-2015 16:30:33	304d 2h 46m 29s	1/2	OK - load average: 0.02, 0.01, 0.01
	Current Users	OK	06-08-2015 16:33:32	304d 2h 43m 42s	1/2	USERS OK - 0 users currently logged in
	Processes	OK	06-08-2015 16:36:04	304d 2h 40m 55s	1/2	PROCS OK: 688 processes with STATE = RSZDT
	Root Partition	OK	06-08-2015 16:39:31	304d 2h 38m 8s	1/2	DISK OK - free space: / 988 MB (51% inode=86%):
	Swap Usage	OK	06-08-2015 16:41:59	304d 2h 35m 21s	1/2	SWAP OK - 100% free (31983 MB out of 31999 MB)
	var Partition	OK	06-08-2015 16:45:00	304d 2h 32m 34s	1/2	DISK OK - free space: /var 3446 MB (90% inode=98%):
virt4	Check All Disks	OK	06-08-2015 16:47:33	304d 2h 29m 47s	1/2	DISK OK - free space: / 1143 MB (58% inode=87%): /var 3353 MB (87% inode=92%): /tmp 1798 MB (91% inode=99%): /usr 2713 MB (70% inode=87%): /dev/shm 32137 MB (99% inode=99%):
	Current Load	OK	06-08-2015 16:49:54	304d 2h 27m 0s	1/2	OK - load average: 0.02, 0.06, 0.06
	Current Users	OK	06-08-2015 16:52:41	304d 2h 24m 13s	1/2	USERS OK - 0 users currently logged in
	Processes	OK	06-08-2015 16:55:40	304d 2h 21m 26s	1/2	PROCS OK: 681 processes with STATE = RSZDT
	Root Partition	OK	06-08-2015 16:27:49	304d 2h 49m 10s	1/2	DISK OK - free space: / 1143 MB (58% inode=87%):
	Swap Usage	OK	06-08-2015 16:30:33	304d 2h 46m 23s	1/2	SWAP OK - 100% free (31999 MB out of 31999 MB)
	var Partition	OK	06-08-2015 16:33:29	304d 2h 43m 36s	1/2	DISK OK - free space: /var 3353 MB (87% inode=92%):
virt5	Current Load	OK	06-08-2015 16:36:11	1003d 19h 54m 58s	1/2	OK - load average: 0.10, 0.11, 0.12
	IPMI	OK	06-08-2015 16:54:46	248d 23h 22m 19s	1/2	IPMI Status: OK
	Processes	OK	06-08-2015 16:42:56	1236d 14h 58m 21s	1/2	PROCS OK: 542 processes with STATE = RSZDT
	RAID	OK	06-08-2015 16:45:10	1220d 22h 10m 35s	1/2	Okay (OKY) RAID1 Optimal (OPT) Optimal (OPT) Hot Spare (HSP) Standby (SBY)
	Root Partition	OK	06-08-2015 16:47:33	1220d 21h 22m 49s	1/2	DISK OK - free space: / 438 MB (22% inode=77%):
	Swap Usage	OK	06-08-2015 16:49:58	1003d 17h 49m 1s	1/2	SWAP OK - 100% free (32755 MB out of 32765 MB)
	var Partition	OK	06-08-2015 16:53:00	1003d 17h 19m 10s	1/2	DISK OK - free space: /var 2580 MB (67% inode=93%):
virt6	Current Load	OK	06-08-2015 16:56:01	1236d 18h 29m 44s	1/2	OK - load average: 0.02, 0.04, 0.05
	Processes	OK	06-08-2015 16:28:31	416d 16h 16m 49s	1/2	PROCS OK: 535 processes with STATE = RSZDT
	RAID	OK	06-08-2015 16:30:40	1236d 18h 21m 29s	1/2	Okay (OKY) RAID1 Optimal (OPT) Optimal (OPT) Hot Spare (HSP) Standby (SBY)
	Root Partition	OK	06-08-2015 16:33:44	382d 14h 56m 15s	1/2	DISK OK - free space: / 944 MB (49% inode=84%):
	Swap Usage	OK	06-08-2015 16:36:18	990d 16h 16m 48s	1/2	SWAP OK - 100% free (32439 MB out of 32765 MB)
	var Partition	OK	06-08-2015 16:39:31	990d 16h 15m 1s	1/2	DISK OK - free space: /var 2857 MB (74% inode=95%):

Figure 4.1.: Nagios instance at WLCG Tier 2 centre GoeGrid.

administrators using the CFengine tool. Advantage of Ganglia monitoring is to provide the hardware components and the system load metrics with different granularity, from hourly and up to yearly length. Therefore monitoring of the GoeGrid infrastructure with Ganglia and Nagios gives advantage of observing and analysing not only the current state of GoeGrid but also looking into the periods of stable and problematic performance.
The Ganglia monitoring tool consists of three main components, the gmond and gmetad daemons and the ganglia-web interface for displaying the aggregated information in an interpretable and useful way [159]. The gmond daemon is deployed on each host under the ganglia monitoring including the head node. It has two main states, such that it can be used for the monitoring information aggregation and transmission to the head node and also for receiving and storing the monitoring metrics from all hosts in the cache memory. The gmetad daemon pulls periodically the cached monitoring data from the memory and writes in the Round Robin Database (RRD), also it supports the User Interface (UI) generation of the Ganglia web frontend. The aggregated data in the RRD is displayed using the web frontend in a general view (see fig. 4.2) and also one can go look for a particular metrics of a single host (see fig. 4.3).

4.3.3. HappyFace

The HappyFace [160] has been initiated as a joint project between two main participants the Karlsruhe Institute of Technology (KIT) and the Georg-August-Universität Göttingen. According to the taxonomy of Grid monitoring tools [161] the HappyFace have the sensors, producer and republisher, hence belongs to the level two monitoring tool. It


GoeGrid_Server Cluster Report for Mon, 08 Jun 2015 17:22:27 +0200

Metric load_one Last hour Sorted descending

Grid > GoeGrid_Server > se-goegrid.local

Overview of GoeGrid_Server

CPUs Total: 254
Hosts up: 21
Hosts down: 0

Avg Load (15, 5, 1m):
5%, 5%, 6%
Localtime:
2015-06-08 17:22

Show Hosts: yes ○ no ● | GoeGrid_Server load_one last hour sorted descending | Columns 5 Size small

Ganglia
.sourceforge.net
Physical View
GoeGrid_Server Cluster Load last hour
Load/Procs
1-min Load
Nodes
CPUs
Running Processes
GoeGrid_Server Cluster CPU last hour
Percent
User CPU
Nice CPU
System CPU
WAIT CPU
Idle CPU
GoeGrid_Server Cluster Memory last hour
Bytes
Memory Used
Memory Shared
Memory Cached
Memory Buffered
Memory Swapped
Total In-Core Memory
GoeGrid_Server Cluster Network last hour
Bytes/sec
In
Out
Cluster Load Percentages
50-75 (4.76%)
25-50 (4.76%)
0-25 (90.48%)
se-goegrid.local
load_one last hour (now 6.67)
virt5.local
load_one last hour (now 6.02)
creamce2.local
load_one last hour (now 0.21)
nagios.local
load_one last hour (now 0.10)
creamce3.local
load_one last hour (now 0.20)
virt6.local
load_one last hour (now 1.16)
virt1.local
load_one last hour (now 0.29)
se16-goegrid.local
load_one last hour (now 0.14)
virt4.local
load_one last hour (now 0.42)
virt3.local
load_one last hour (now 0.26)
se14-goegrid.local
load_one last hour (now 0.05)
se12-goegrid.local
load_one last hour (now 0.03)

Figure 4.2.: General view of Ganglia monitoring tool at GoeGrid.

has a single point of access, provides actual, analysed data displayed using the simple ranking mechanism, provides history functionality to view the status of the monitored services or facility components in the past at a given date and time and the tool has a modular structure. Each module of the HappyFace tool allows to collect various crucial monitoring and performance information about the particular site. Due to the flexibility of modules, it is possible to fetch the information published in a different formats from various monitoring and accounting sources, thus the tool does not require deployment of its own clients on various hosts and stands on top of existing monitoring applications as a meta-monitoring framework.
The current version of HappyFace (version 3) is completely based on *Python* programming language [162] and the dynamic web framework *CherryPy* [163]. Functionality of HappyFace is based on its core scripts, which needs to be executed subsequentially. Initially the `acquire.py` should be launched, it reads all the configuration files and runs all the modules in order to fetch and store the fresh information from various sources. The next is a `render.py` script, which access the last entry of aggregated and stored data in the database and process it to generate the output for the HappyFace web frontend (see fig. 4.4).

At GoeGrid, the HappyFace is used to aggregate the monitoring information from internal monitoring tools such as local instances of Nagios and Ganglia, also it collects the site related monitoring data from external sources such as central ADC monitoring tools like Distributed Data Management (DDM) Dashboard, PanDA system etc. Since the GoeGrid is the computing and storage resource provider for ADC and so far it is the only ATLAS site using the HappyFace meta-monitoring tool, most of the modules

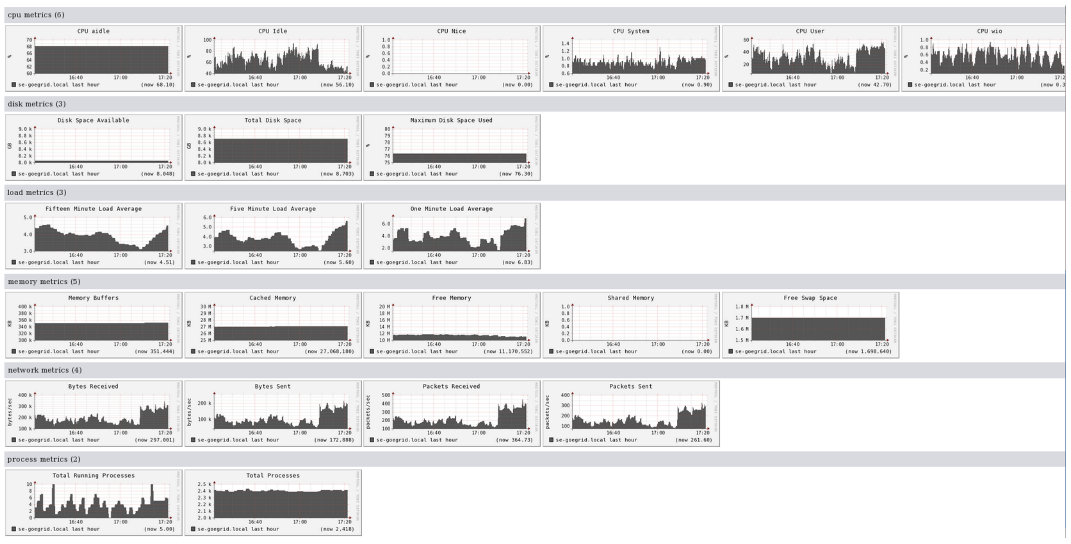
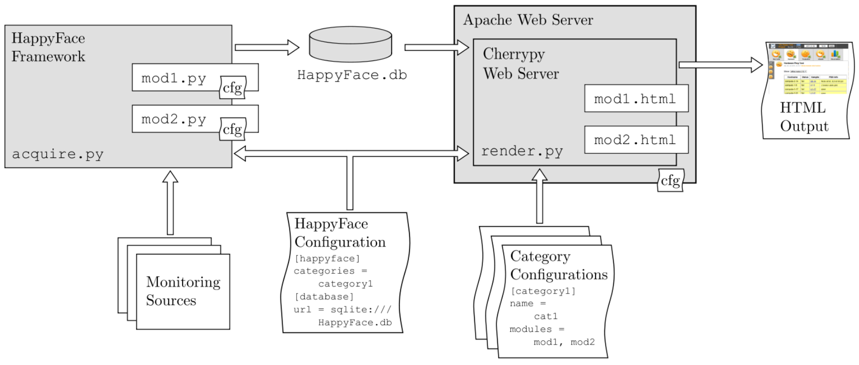


Figure 4.3.: Monitoring metrics of a particular host view at GoeGrid Ganglia instance.



```
[happyface]
categories =
    category1
[database]
url = sqlite:///
    HappyFace.db
```

```
[category1]
name =
    cat1
modules =
    mod1, mod2
```

Figure 4.4.: The HappyFace meta-monitoring tool workflow.

collecting information from the central monitoring tools were developed and deployed locally, at the II Physics Institute of Georg-August-Universität Göttingen.
The HappyFace meta-monitoring application groups the modules output in categories. Up to now, GoeGrid instance of HappyFace [164] has six categories, where one of them ”DB Webservicers” is dedicated for provisioning of the aggregated data for further analysis. The first category, ”Monitoring” displays the GoeGrid critical service test results from the Service Availability Monitoring portal [165]. The second module in the same category aggregates the monitoring information from the GoeGrid Nagios instance and displays it in the table. The ”Site Services” category also contains two modules dedicated for checking the correctness of published information from GoeGrid Apel and BDII instances. Also, the modules, check if all the information is up to date and in case of violation of the predefined thresholds, written in the corresponding configuration files,

the alarm color code, "red", is used on the web frontend. In the category "dCache" also two modules, the "dCache Pool Information" and the "dCache Dataset Restore Monitor (Lazy)" show details of the current status of GoeGrid dCache storage management system and its key components. The category "PanDA Info" contains four modules, showing efficiency and status information about the GoeGrid production and analysis queues in the PanDA system. Also the status of the functional tests, sent to GoeGrid PanDA queues, to check its availability and reliability. The detailed information about the Ganga jobs efficiency, submitted to the GoeGrid analysis queue, together with the failed jobs statistics per queue and worker nodes, makes this category very useful for the site administrators. The final category "DDM" shows the efficiency of data movement related activities. In particular two modules show the efficiency of the GoeGrid storage as a source and as a destination for the data transfers and the data deletion efficiency.
All GoeGrid management and monitoring tools as well as the strategy for the GoeGrid administration proved its efficiency during the period of the site operation as an Tier 2 centre, i.e. since 2008. Up to now the GoeGrid efficiency in terms of reliability and availability metrics is constantly over 95%.

5 Autonomic Computing

During the last decade worldwide coverage of the internet has increased dramatically together with the bandwidth capabilities of the network infrastructure. Average speed of the internet access world-wide is in the range of several megabits per second. High speed interconnection as well as the wide network coverage of the most populated parts of the world has naturally increased demand on the digital services. Moreover the number of provided digital services is also growing almost on a daily bases. These drastic changes were caused due to the fast technological growth, where leading role has the revolution in mobile technologies and communications. Most of the digital activities, which nowadays take place in everyday life, can be performed on the mobile devices, which increases the time of being connected to a particular social or media networks as well as to the digital commerce and communication applications or service providers.
Growing demand on digital services should be responded by the corresponding, stable supply. Stability and availability of the provided services is the key requirement for the digital service suppliers. Nowadays it is not possible to imagine that online shopping, social media, communication applications or cloud platforms are not available for even a short period of time. It is, first of all, harmful for the digital companies which will loose reasonable amount of revenue on a minute scale. Therefore burden to handle and provide continuous services goes on the system administrators. The amount of tasks and work is only growing and there is no tendency to assume that it will reach certain point of saturation. This fact provokes need for an automation of certain management and administration processes performed by the human system administrators.
This is not the first time when the technological evolution requires the general solution in terms of the management automation. Similar problem arose in the beginning of the twentieth century, when the human operators were working on the manual switchboards in order to connect two users of the telephone network. The number of users increased dramatically and there was a lack of trained operators for supporting the telecommu-

nication [166]. This problem, which could affect the entire evolution of the telecommunications as a field, has been solved by the invention and deployment of the automatic branch exchange. Despite to the simplicity of the problem of telephone communication management automation, in case of the digital service provisioning there is a problem set, which is broader and harder to handle. Complexity of the challenge is mainly due to the big number of critical, independent subcomponents, which are necessary for provisioning of a certain digital services. Each of these subcomponents independently have to be stable in order to guarantee the overall system availability, reliability and serviceability.

Automation of the complex system management, with the big number of independent complex subcomponents, is a known topic of research for the last several decades. Before arising of this problem, for the complex digital service providers, there were several significant projects in related fields dealing with the similar challenges. The first well-known project was initiated by the *Defence Advanced Research Projects Agency* DARPA [167] in 1997. The main goal of the project was automation of the communication management among the soldiers spread in different locations of the battlefield. The core idea was provisioning of the information from a particular soldier to the entire group, about the health condition and the certain characteristics of the location. Analysis of all the information provided from the group of soldiers would allow a proper and efficient planning and assessment of a particular mission status and progress, which plays critical role for achievement of success with small amount of loss. Together with critical information provisioning and analysis, important aspect was the security of all communication channels for avoiding any kind of intrusion. In order to solve this very complex problem there was suggested an multi-hop ad-hoc routing [168] approach, i.e. the signal from a soldier is sent to the nearest neighbour, afterwords the signal receiver sends it to the next nearest neighbour, until all the group members are covered. This is known to be an decentralized peer-to-peer mobile adaptive routing, which can be identified as a problem of the complex system management automation. Another project, also initiated in terms of DARPA in 2000, was the DASADA [169]. The goal of this project was more related to the problem of the management processes automation of the modern digital service providers, than in case of the 1997 DARPA initiative. In terms of DASADA project the reusable and dynamic software system was developed, with an ability of self-adaptation to the different changes of external or internal environmental conditions. This project gave the significant contribution in terms of systematic approach development for the complex system management automation, in particular the architecture-base system development was proposed. Also, basic probes and gauges, for the system monitoring were defined, which serve as a key performance and state indication metrics for the adaptation engine [170–172]. The adaptation engine is the component analysing the current state of the system and its subsystems. Only in 2001, there was a first proposal from IBM [173] suggesting the general approach for the complex digital service providers management automation. The core idea was based on autonomic properties i.e. the system should be able to identify and solve the problems without the involvement of human, which eventually would lead to the decrease of the system administrator workload, hence increase

of the entire system availability and reliability. The fourth [174] key project related to the complex system management automation was also initiated by DARPA in 2004. The goal of the program was provisioning of the dynamically adaptable, computing systems providing the critical functionality continuously, despite to attacks or infrastructural damage.
All, four projects share the common feature, the self-management, i.e. the main requirement to the system was the automatic adaptation to the changes in the dynamic environment, where the system is running. The major work from IBM, with the description of the Autonomic Computing (AC) concept, the projects initiated from DARPA and other activities in the direction of the complex systems management automation provoked the development of new concepts, strategies and approaches in the direction of development of self-managing systems. Properties, architecture and implementations of such systems for cloud and grid computing environments will be described in the next sections of this chapter.

5.1. Introduction

Main motivation for development of the Autonomic Computing concept is provisioning of highly available, reliable, secure and serviceable, self-managing computing system. In order to develop and afterwords deploy the computing infrastructure with the options of self-management, i.e. high level of automation, the key requirements need to be defined. According to [175] the system with the self-management features, should be able to recognize and analyse its own state and status, which also means understanding and analysis of its subsystems conditions. This feature can be considered as a self-awareness allowing automatic identification and reaction on the critical situations. The first action, when the critical situation is observed, is the initiation, of the problem root cause analysis (RCA) process. The goal of the RCA is the detection of a particular problematic component of the entire infrastructure and its isolation in order to decrease the malfunction rate for the entire system. The RCA can be considered as a part of self-awareness, while the isolation and fixing of the entire system's problematic part is identified as a self-healing property of the autonomic computing.
The computing facility cannot exist in an isolated environment due to the fact that most of the infrastructures and platforms provide their services to the users via the internet. The internet is an extremely dynamic and rapidly expanding environment where the dangerous activities also take place. The biggest threat from the internet to the large digital service providers are the multiple and continuous attacks that cannot be quickly and easily distinguished from the users requests, especially if the assault rate is low. Automatic detection of the attack, as well as its prevention and adequate reaction on it is additional essential and required component of the autonomic computing concept, which is called the self-protection. It is impossible to substitute the system and security administrators with an automatic systems, with the self-protection features, completely but existence of such components is an opportunity to lighten the burden on the personnel, which also allows to concentrate the system administrators skills and knowledge

on the analytical tasks related to the systems and infrastructure security.
The hardware component of the large computing platforms does not remain static for the long periods of time. Frequently there is a need of hardware component exchange, or the entire infrastructure expansion as well as decommissioning of the outdated hardware. After all these actions the system based on the modified hardware needs to be adapted to the changed conditions. This task mainly relies on the system administrators and engineers but concept of the autonomic computing also covers such an activities. The self-configuration feature of AC covers the aspect of the automatic system adaptation to the changes on the hardware level.
Large digital service providers usually have the distributed structure and are based on a complex software systems which have a number of tuning parameters. Proper adjustment of all or at least the key parameters of various systems is the most important factor for assurance of high service availability. Increasing complexity in terms of the number of tuning parameters and the layers used in the modern software systems requires bigger number of skilled and experienced personnel, which is not always available. Automation of the systems adjustment plays a key role in further efficient usage and expansion of the complex software systems providing the number of modern digital services. The self-optimization is the feature of an AC, which is meant for handling the software systems tuning and adjustment for achievement the high service availability, reliability and serviceability.
The general properties of an autonomic computing the self-configuration, self-healing, self-optimization and self-protection in practice are associated to the particular actions usually taken by the IT professionals during their daily activities. The most of these actions are repetitive and are taken several times during the system administrator's working time. This circumferences allows to automate number of such an activities named as control loops. Large digital service providers follow the best practices and standards of *IT Infrastructure Library* the ITIL [176], hence there are defined the control loops, for most of the activities and incidents, taking place inside the infrastructure. Examples of such an activities are "incident management", "problem management" or "change management" etc., for all these activities the corresponding workflow is defined, which needs to be accomplished in order to reach the final general objective. The objectives are defined by the IT professionals and therefore when considering an automation of the control loops it is important to have the corresponding interface where mostly the system administrators will be able to identify the policies and general goals. Thus together with the four general properties the requirement to AC is to have the corresponding communication mechanism mostly for the system administrators and IT professionals.
According to the requirements and needs for the automation of the large scale computing facility management processes there can be identified key components that needs to be managed, which are different from each other by their functionality and characteristics. The complex digital service provider platforms consist of a number of hardware and virtual servers, network equipment and the software applications. All of these entities needs to be protected, managed, configured and healed, hence it is important to run the control loop on the level of each component of the infrastructure as well as on the

level of orchestrating them as a single entity. Thus, the autonomic computing can be considered as a complex system with the layered structure (see fig. 5.1) [177].
The "Managed Resources" layer is similar to the grid fabric layer (see fig. 3.2), its

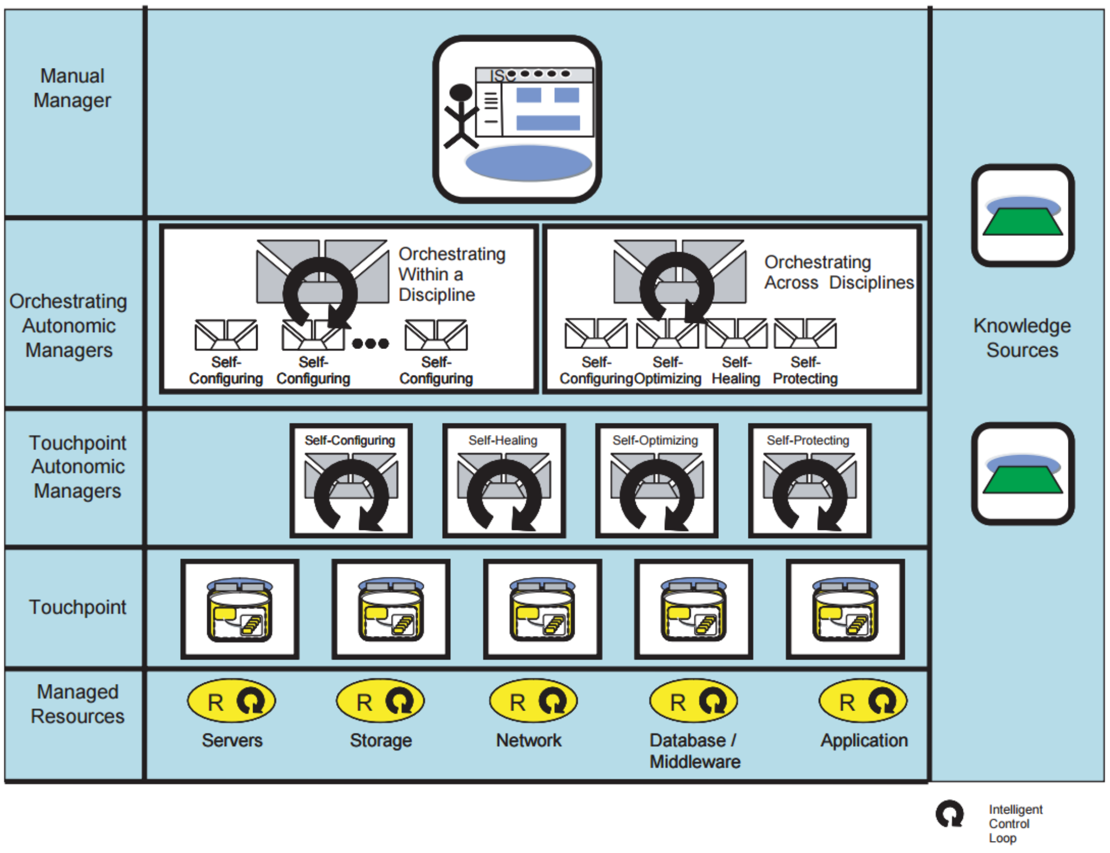


Figure 5.1.: The layered structure of Autonomic Computing according to [177].

components are hardware and virtual servers, applications and application servers, software and hardware constituents of the network infrastructure as well as the database application servers. The "Managed Resources" components are managed by own self-management control loop together with the autonomic managers from the upper layer. The intelligent control loop for each element of the "Managed Resource" layer is integrated deeply in the run-time environment in order to guarantee the self-management functionality for the very specific components of a particular server, service or application.
The "Touchpoints" implements the key functional components of the self-managed system, the sensors and effectors. The "Touchpoints" communicate with the managed resources via the sensors in order to analyse its configuration files and log data and also to perform the detailed, monitoring data aggregation to know the state and the condition of a resource itself and its sub-components. The effectors are meant for action

taking on the resource level in order to avoid its degradation or failure.
The large digital service providers consist of plethora of "Managed Resources" components and most of them are SOA based. This fact assumes that there is a communication with and within the resources affecting positively or negatively on entire infrastructure availability and serviceability. In order to guarantee the overall high performance of the digital service provider platform the self-management ability of a particular component is not enough, the high-level governance of the "Managed Resources" intelligent control loops is required. The "Touchpoint Autonomic Managers" layer guarantees the orchestration of a single or group of "Touchpoints" in order to achieve the reliable provisioning of services to the consumers without violation of the SLA conditions. According to [177] there can be identified four scoped of resources that can be managed by the "Touchpoin Autonomic Manager" on top.

- The first scope of components covers the single entities, like network switches, routers or single servers and services. These are the building blocks of the entire infrastructure and orchestrating of each of the entity has a key value for entire infrastructure.
- In the large computing facility it is possible to group the components according to the similarity of the provided services or according to the architecture. Components grouped by the certain criteria of similarity forms the homogeneous group, which is the second scope that needs to be dynamically optimize in order to gain the maximal throughput from the facility.
- Unity of heterogeneous devices and servers working to achieve the higher level goal forms the third scope, where the higher level orchestrating is required. Example of such a heterogeneous system is the unity of web and database servers together with the storage management system, which provide the complete functionality for storing, modification and management of the data on the remote locations. This combination of heterogeneous systems is widely used by the cloud platforms providing the storage services.
- The last realm of activities where the "Touchpoint Autonomic Managers" are essential is the business services. This can be considered as the system with the highest complexities, where the policies for the service delivery to the consumers are defined based on business processes, that needs to be converted in the language of the first scope systems performance optimization.

The "Touchpoints" and "Touchpoint Autonomic Managers" does not cover the entire infrastructure as a common entity. In order to perform the complete implementation of the self-managed system it is important to have the high level managers, which coordinate activities of low level components. In case of the suggested layered structure, "Orchestrating Autonomic Managers" component has this role. The orchestrating managers are classified in two major groups, *within a discipline* and *across the discipline.*

The *within a discipline* managers coordinate the number of homogeneous touchpoint managers i.e. managers with the identical self-management features.
The *across the discipline* managers coordinate among the heterogeneous touchpoint managers i.e. handling different types of self-management properties.
The "Manual Managers" is the component of an AC, which mainly provides the interface to the IT professionals and system administrators for definition, checking and providing their policies and knowledge in AC acceptable manner.
The most important component of the layered structure is the knowledge. Without system related knowledge acquisition it is impossible to consider the self-awareness, hence other key features of AC. The general approach for the automatic knowledge acquisition is discussed in the following section of this chapter.
The proposed layered structure also satisfies the key elements of AC as a discipline, which were defined by the IBM researches [178] in 2001.

1. "To be autonomic, a computing system needs to *know itself* - and comprise components that also possess a system identity", i.e. self-awareness.

2. "An autonomic computing system must configure and reconfigure itself under varying and unpredictable conditions". This key requirement can be separated in two major parts: The system should be able to anticipate the problems and changes in an extremely dynamic environment as well as continuously adapt itself to present internal and external changes to achieve high performance and reliability of the provided services.

3. "An autonomic computing system never settles for the status quo - it always looks for ways to optimize its workings". In order to fulfil the functionality required for realization of this key requirement self-awareness in conjunction with the efficient execution of the control loops is essential.

4. "An autonomic computing system must perform something skin to healing - it must be able to recover from routine and extraordinary events that might cause some of its parts to malfunction". First of all, this key element leads to awareness of avoiding two major risks for the large scale computing platforms: Cascade failing of the entire infrastructure in case of a single component degradation and need of damaged component isolation. Among the infrastructure it is not acceptable to have a single instance of a critical service, which can lead of the entire system degradation in case of its damage or harming. Isolation of the affected component also plays key role in terms of understanding the weak points of the infrastructure security and avoiding further spreading of the service come-down. This aspect is mainly handled by the self-healing and self-protection features of AC.

5. "A virtual world is no less dangerous that the physical one, so an autonomic computing system must be an expert in self-protection". For realisation of this feature implementing of self-protection properties needs to be considered.

6. "An autonomic computing system knows its environment and the context surroundings its activity, and acts accordingly". This key requirement mainly refers to the all components of the layered structure and also to all four main properties of AC.

7. "An autonomic computing system cannot exist in a hermetic environment". Large digital service provider platforms are open to their consumers via the internet, the same is true for the scientific grid computing service users. This aspect is considered in terms of self-awareness, self-healing and self-protection properties.

8. "Perhaps most critical for the user, an autonomic computing system will anticipate the optimized resources needed while keeping its complexity hidden". This feature points to the core component of AC - the knowledge about system or self-awareness property. Number of methods, including full automated or semi-automated has been developed in terms of AC projects. So far there is no complete knowledge acquisition system that aggregates all the information in terms of all four key features of AC but there are mechanisms able to accomplish this task for various aspects. The research in this direction is the hot topic of research in the field of Autonomic Computing.

Clear definition of the structure of AC together with provisioning of the discipline key elements allows to discuss the functional aspects of the autonomic computing architecture. The AC deployment specifications and details, for HPC and HTC environments, are provided in the next section as well as the specification of autonomic levels.

5.2. The Core Components

Stable performance of any complex system, like large distributed computing platforms or alive organisms, depend on large number of factors, which can be classified as internal and external. In order to keep the complex system stability, it is important to guarantee that variables characterizing the factors are kept in the range of certain limits. The key aspect of the autonomic nervous system is exactly to keep the organism's critical parameters in the viable zone. This is achieved by the constant adaptation of body's sub-systems to the dynamic changes. According to Ashby [179], as a results of constant adaptation, i.e. adaptive behaviour the key variables, indicating the system's overall stability, remain in their physiological limits. These limits are used to define the organism's viability zone. Thus, the autonomic nervous system main functionality is to keep the organism in the state of equilibrium by adapting its subcomponent parameters to remain in their physiological limits.
Based on observations, Ashby [179] also classified two types of disturbing factors, causing the organism equilibrium state violation, hence motivating the constant adaptation processes. The small continuous signals, affecting the main variables without leaving the viability zone, belongs to the first group of disturbing factors. The second group of incidents, affecting the system stability, are causing the significant changes, hence

functionality of affected subcomponents goes beyond the equilibrium state. In order to handle the effects caused by the disturbing factors from the first group it is not required big effort from the adaptation mechanism, but disturbances from the second group requires stronger response, which also can take more time e.g. fever as a reaction of body to fight against the infection take last from several hours to several days. All these observations motivated Ashby to identify the architecture of ultra-stable system fig. 5.2. Any complex system exists in a certain environment, which serves as a medium

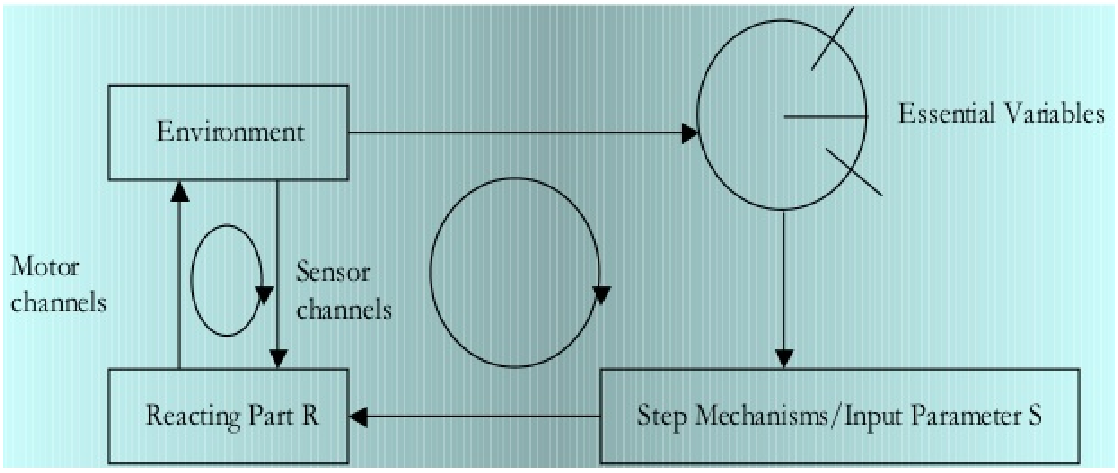


Figure 5.2.: Ashby's Ultra Stable System Architecture [179].

for passing the external disturbing factors. The "R" component of the ultra stable system communicates with the environment using the "Sensor Channels" and based on the aggregated information performs the stabilizing actions using the "Motor Channels". If the disturbing influence is strong enough to cause the major shift in the "Essential Variables" characterising the system stability, the result will be violation of the system's equilibrium state, hence the "S" component of the model will trigger actions from the "R" component. The system stabilization using the "R" component is called a small control loop, while the workflow including the "S" component identifies the big control loop of actions.
As it is shown in [180], the human nervous system, hence the autonomic nervous system has the adaptive features and can be considered as an Ashby's ultra stable system. Since the baseline model for the autonomic computing is an autonomic nervous system, it should contain all the components of the Ashby's model, in order to guarantee that all the sub-components, hence the entire system will remain in equilibrium state.
The conceptual structure of an autonomic computing architecture is presented on the scheme 5.3. In this case challenges for High Performance Computing (HPC) and High Throughput Computing (HTC) are similar, therefore the same concept can be efficiently implemented in grid and cloud computing environments.

On the scheme environment contains all the internal and external factors that influence the computing platform performance and stability. The internal factors can affect

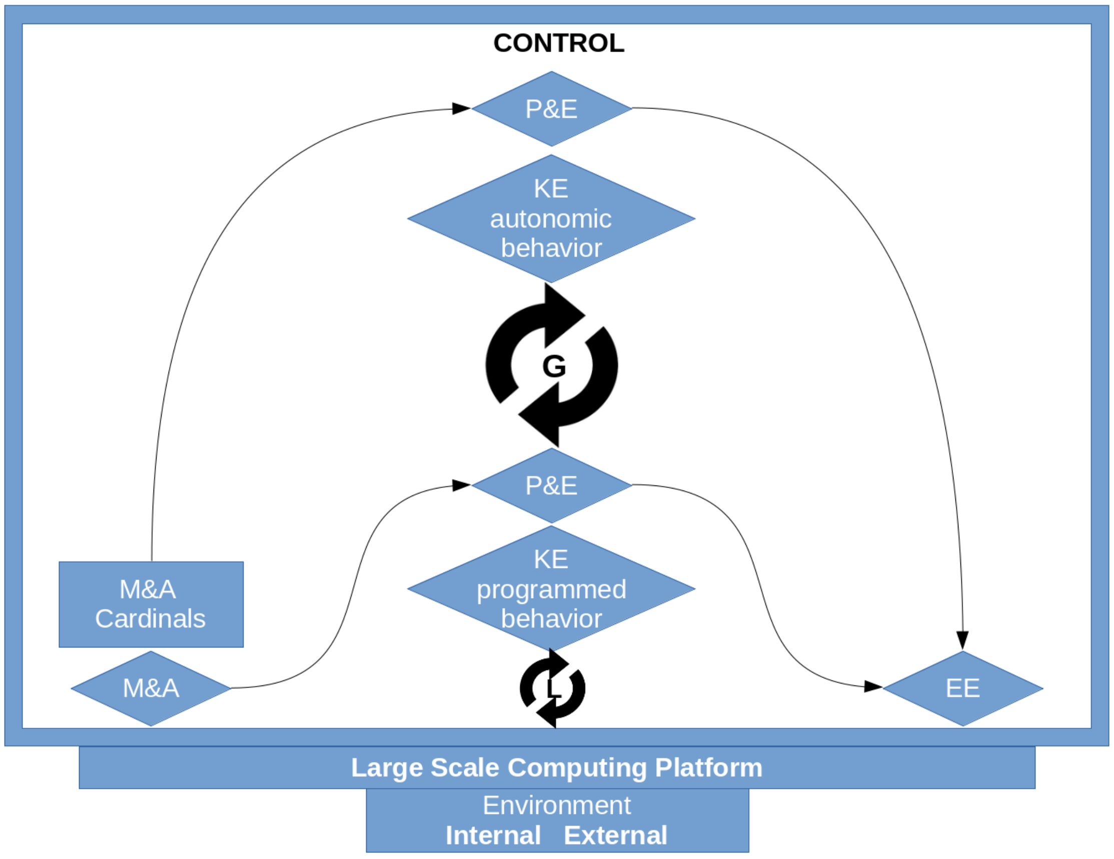
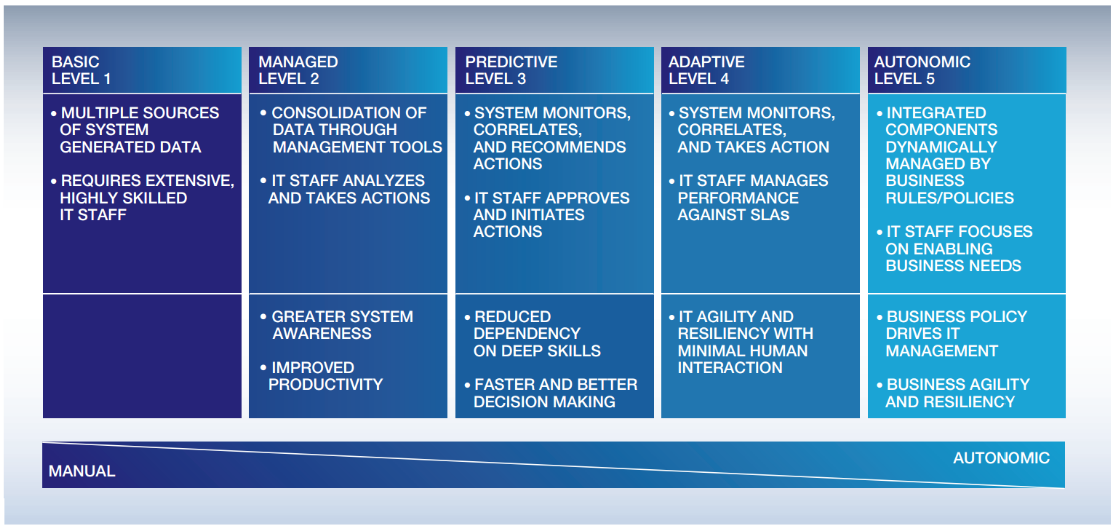


Figure 5.3.: Conceptual architecture of autonomic computing system.

the digital infrastructure on the the application runtime level, while deviations of the external factors influence the application execution environment. Effect caused by the both types of factors is handled by the "Control" system of an autonomic computing. The key components of the control are the Monitoring and Analysis Engine (M&A), Planning Engine (P&E), Knowledge Engine (KE) and an Execution Engine (EE).
The Monitoring and Analysis Engine (M&A) continuously follows the changes and performance of the computing system using the sensors. The aggregated information is analysed and processed for its further application and visualisation, that helps the IT professionals and the system administrators in problem root identification.
The Planning Engine (P&E) generates an optimal set of actions in order to tune the computing infrastructure performance by means of self-healing, self-optimization, self-protection and self-configuration optimization.
The Knowledge Engine (KE) contains the policies for various critical and performance-harming cases. It accumulates the experience of solving the problems either using the automatic knowledge acquisition systems or by the IT experts. The aggregated knowledge base serves as a guideline for identification of the most relevant plan of actions

when dealing with a particular problem.
The Execution Engine (EE) implements approved action plans and supports the computing infrastructure adaptation to the dynamic environment affected by the number of internal and external disturbing factors.
For efficient implementation of all the control components the functional constituents are necessary. On the scheme 5.3 two, the local (L) and global (G), control loops server as functional parts of the entire control mechanism.
The local control loop (L) exists to check the entire infrastructure's sub-systems performance and stabilize their functionality. This component of an autonomic computing architecture is similar to the "R" component of Ashby's ultra-stable system, which only reacts on the small disturbances but does not take into account the overall state of the entire computing platform. The current status of all the computing platform's subcomponents are analysed based on the data provided by the M&A component. Based on aggregated monitoring information and the knowledge accumulated at K&E component the set of anticipated problems are defined with the list of corresponding actions. Once the sequence of relevant actions is identified the local control loop executes them using the Execution Engine. The mapping among the data, provided from sensors, and the computing platform sub-systems can be defined by an IT experts or using the set of automatic knowledge acquisition methods.
In order to fill the functional gap of the local control system in terms of the conceptual architecture of the autonomic computing model the global control loop (G) mechanism is introduced. The global control loop, has similar functionality as the "S" component of the Ashby's system, it triggers actions for adaptation to the environmental changes caused by the strong disturbances. The changes on the environmental level are registered based on analysis of the monitoring data provided from the M&A component. In particular the performance, fault-tolerance, configuration and the security aspects of the entire system are under constant observation and check to identify and if possible predict the environmental shift. If the changes affecting the entire distributed computing platform are identified the global loop comes to actions and the first step is to analyse the possible patterns of actions in order to guarantee the system's availability, reliability and serviceability. The patterns of actions and analysis of their consistency to the particular problematic scenario is performed using the Knowledge Engine (KW) component. On the next step the Planing Engine (P&E) provides the sequence of steps to the Execution Engine (EE), which finally performs the entire adaptation process including the adequate configuration of each sub-system.
Based on these components of the autonomic computing conceptual model, the main self-* properties of the system is guaranteed. However, it is not always needed or possible to implement the entire architecture completely, hence the levels of system autonomy can be distinguished. According to [181], there are identified five levels of large computing platform autonomy (see fig. 5.4).

1. The initial level of autonomy does not assume implementation of the autonomic computing system. Computing platforms can be at this level of autonomy if mostly



BASIC LEVEL 1	MANAGED LEVEL 2	PREDICTIVE LEVEL 3	ADAPTIVE LEVEL 4	AUTONOMIC LEVEL 5
• MULTIPLE SOURCES OF SYSTEM GENERATED DATA • REQUIRES EXTENSIVE, HIGHLY SKILLED IT STAFF	• CONSOLIDATION OF DATA THROUGH MANAGEMENT TOOLS • IT STAFF ANALYZES AND TAKES ACTIONS	• SYSTEM MONITORS, CORRELATES, AND RECOMMENDS ACTIONS • IT STAFF APPROVES AND INITIATES ACTIONS	• SYSTEM MONITORS, CORRELATES, AND TAKES ACTION • IT STAFF MANAGES PERFORMANCE AGAINST SLAs	• INTEGRATED COMPONENTS DYNAMICALLY MANAGED BY BUSINESS RULES/POLICIES • IT STAFF FOCUSES ON ENABLING BUSINESS NEEDS
	• GREATER SYSTEM AWARENESS • IMPROVED PRODUCTIVITY	• REDUCED DEPENDENCY ON DEEP SKILLS • FASTER AND BETTER DECISION MAKING	• IT AGILITY AND RESILIENCY WITH MINIMAL HUMAN INTERACTION	• BUSINESS POLICY DRIVES IT MANAGEMENT • BUSINESS AGILITY AND RESILIENCY

Figure 5.4.: Levels of the large computing platform autonomy [181].

the processes are managed by the IT professionals and experienced system administrators.

2. At the second level of autonomy, system administrators and the infrastructure responsible personnel rely on the monitoring and alarming tools deployed inside the platform. Still at this level it is not considered to have the integrated monitoring or automation tools.

3. The third level systems, together with the system performance and stability monitoring tools contain the analytical capabilities and providing recommendations for running particular actions corresponding to the problematic or warning state.

4. The fourth level of automation, assumes better implementation of an systems adaptive capabilities. The autonomic system accepts the high level policies and goals defined by the IT professionals and attempts to follow them in conjunction with the system administrators. However the final decisions for most of the actions are still up to the human experts.

5. The systems of fifth level or an autonomic level are able to perform the local and global control loops, i.e. adapt themselves to the changing environment as well as give a response to the local disturbing factors.

The number of systems and projects were emerged during the last decade to achieve the fifth level of autonomy but still, development of such a system is considered to be a big challenge. Contribution in development of particular components of an autonomic computing architecture, plays significant role in overall evolution of this field. Especially it is motivated by participating of big IT companies such as IBM, HP and Microsoft. These companies already are providing systems with different levels of autonomy.

5.3. Autonomic Computing Implementations

Since the publication of the first Autonomic Computing blueprint by the IBM in 2003 [177], the development of its different components as well as the entire system has been triggered. For avoiding the terminology duplication as well as wasting resources on the identical developments IBM proposed the common terminology and afterwords started active collaboration with the various grid computing communities e.g. OGSA in order to integrate an Autonomic Computing components in grid standards.
In parallel of the initiating the projects related to the standardization and integration of autonomic computing, IBM provided pioneering automated components with automatic and efficient workload management, monitoring, configuration and protection capabilities.
In the activities initiated by IBM, the HP and the Microsoft were involved soon, which led to the development of number of systems with self-managed features and components. One of the first system, the Tivoli [182], with autonomic capabilities was developed by IBM. The goal of Tivoli system was provisioning of self-adaptation features to the complex and heterogeneous IT systems.

Tivoli is not a monolithic system, it consists of number of subcomponents (see fig. 5.5). The constituents allow the system consumers to follow the large computing platform availability and its performance parameters in general. Also it guarantees the self-protection capabilities in terms of its storage system. Tivoli storage manager efficiently performs the data backup procedures and minimize the data-loss risks. The self-optimization feature of Tivoli is integrated in its workload management system, which optimize the software and hardware usage in order to meet the SLA requirements. The self-configuration property is also a part of Tivoli system and is mainly handled by the monitoring and event correlation tools. These tools are using the sensors and based on acquired information are able to identify the changes in the entire infrastructure.
Automatic healing functionality is also integrated in Tivoli mainly in several components, such as Tivoli Switch Analyser. This component analyses the network infrastructure of the computing platform, also is able to detect the problem, its root cause and take the corresponding healing actions. Another component with the self-healing capabilities is the Tivoli Storage Resource Manager, which checks the validity of the predefined policies in the context of system and in case of inconsistency the storage allocation quotas are adjusted in order to exclude any data loss.
Despite to the general purpose autonomic computing software Tivoli, the grid computing systems differ from each other according to its scientific, business or social application domain. The architecture and the component structure also are unique for various grid computing platforms. However, implementation of self-management features as functional components of the grid systems is essential for provisioning the high level efficiency, availability, reliability and serviceability from the available resources.
In the following part of this subsection, the grid computing subsystems and constituents, with the self-adaptation and self-management features, are discussed.
The Condor-G [184] uses benefits from Condor and Globus projects, hence provides

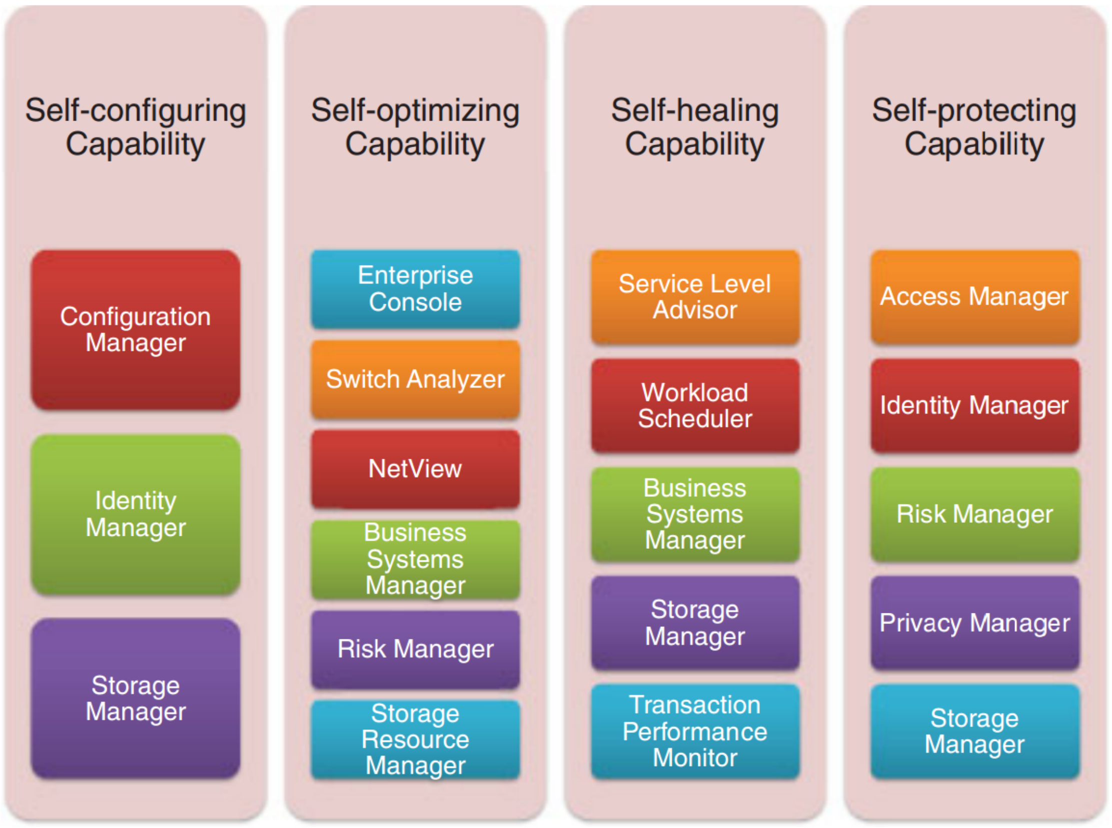


Figure 5.5.: IBM Tivoli Components [183].

grid and local computing and job resource management functionality. The grid level resources and security management is enabled using the Globus tools, while the job management at the batch system level is organized via the Condor components. Computation management service of Condor-G is called Condor-G agent. It provides API and CLI to the end-users such that the entire distributed and heterogeneous computing platform can be accessed and treated as a local batch system. Using the Condor-G user interfaces it is possible to submit the job by providing the name of the executable code, input and output data entities and their names and also the arguments describing the resources on demand. The Condor-G also allows to follow the submitted job status and to cancel it. It also constantly keeps informed the job submitter about the jobs status or its unexpected termination via the callbacks or email notifications, also it provides the access to the jobs detailed log information.
The Condor-G can efficiently manage thousands of jobs with non-trivial resource requirements. Its Direct Acyclic Graph Manager (DAGMan [185]), used as a meta-scheduler, allows to analyse the job execution and runtime monitoring information and in case of resource related problems it resubmits the tasks to another location. This option can be considered as a self-healing property of the system. The self-protection features are

available on the level of Globus constituents.
The main mechanism of realization of the self-* features of Condor-G is the Grid Manager daemon, which is initiated for each submitted job and stopped once the jobs are finished or terminated. Grid manager communicates with each of the daemons in order to dynamically collect the job related information and in case of problems resubmit or reschedule it to another resources according to the predefined policies.
Another example of grid resource and workload manager with an autonomic features is the Nimrod-G software [39]. It is a Globus based grid middleware actively used for deployment of the grid computing sites dedicated for both, business and science.
The core components of the system are, the User station (client application), Parametric Engine, Dispatcher and Job-Wrapper. The web-based UI allows to use the Nimrod-G functionality without installation of the client software. Based on its core components the system is able to perform the self-awareness analysis, self-optimization using the dynamic rescheduling functionality and also self-healing using the failure elimination by its root cause detection. Thus the Nimrod-G can be considered as a self-managing software.
User station component of the system allows users to specify the job parameters like its execution time or cost. It helps to identify the most efficient scheduling decisions as well as to optimize the resource selection process. It also provides the monitoring functionality to the users in order to trace the job status and problematic states.
The Parametric Engine has the similar functionality as the grid workload management system. It holds up to date information about the available resources and their status. Also it interacts with the Nimrod-G scheduler, the clients and guarantees the job runtime environment creation, its maintenance and status tracking functionality. The parametric Engine also makes possible to define the workflow, using the Declarative Parametric Modelling Language (DPML), for entire grid environment.
The Scheduler communicates with the grid information discovery system (MDS for Globus) in order to identify the available computational resources and their status. After the resource discovery, jobs are assigned according to the computational economy-based scheduling strategy, i.e. for the resource allocation, two factors are taken into account the job execution and completion time and the cost of the requested resources.
The Dispatcher functionality is to launch the task execution process of a particular job. The job-wrapper provides the job with the data and also handles the sending back of results.
Advantage of Nimrod-G is ability of efficient functioning in heterogeneous grid environment, e.g. if the corresponding job dispatcher is implemented, the Nimrod-G can be integrated within the Legion [186] and Condor infrastructures.
Taverna [187] is a workload management software within the myGrid project [188], mainly used for the distributed computing systems dedicated for the bioinformatics community. The distinguished feature of Taverna in comparison to Condor-G is handling the processing and transferring the large amounts of data.
Taverna consists of two major parts, the Scufl Workbench application and Freefluo enactor. The Scufl Workbench has GUI for the workflow definition by the system users, while

Freefluo enables communication with the web services and is oriented on the functional part of the workload management system.
In order to define the valid workflow, user is not obliged to know the Scufl (Simple Conceptual Unified Flow language), it is enough to use the corresponding GUI and build the workflow from the provided convenient graphical components. Later the Taverna system translates the graphical workflow in Scufl and enables it for parsing by the Freefluo enactor.
The Freefluo is an workflow managemetn tool developed using the Java technologies. It enables communication with the services such as Soap-lab bio-services, WSDL-based web-services [189] and local applications. Taverna also allows to follow the workflow execution process and provides information about the intermediate state of the ongoing process.
As a part of self-awareness, self-healing and self-optimization properties Taverna implements a number of workflow configuration parameters, e.g. number of job or task execution retries, time delay and altering the processor options. The processor in terms of the Tarverna system corresponds to a particular task. Users are able to define the retry and time delay limits for each of the processor and if after all retries and delay time expiration the task will not be completed it will get the status critical and the submitter will be notified. However, in terms of self-healing and self-configuration features of the system it is possible to follow the task execution process and get the notification immediately once the problem will appear.
One of the largest and stable grid computing platforms with the autonomic capabilities is the WLCG and in particular the ATLAS Distributed Computing (ADC). The PanDA workload management system integrates the pilot and hammer cloud systems, described in the Chapter 3, which guarantees the resources availability and the computing facility reliability. From the autonomic computing view this systems implement the self-awareness, self-monitoring and self-healing properties. Due to the fact that ADC contains sufficient resources, once the hammer cloud system discovers the faulty computing site it will be excluded from the activities until the tests are successful, which can be considered as the implementation of the faulty resources isolation. The similar system works for the ADC data management system. The constant testing of the ADC site storage systems and implementation of the FAX technologies allows the access to the demanded datasets or files even if the actual computing facility, where the jobs are running, is not able to provide the data. The self-configuration component at the ADC is implemented on the level of ADC grid information system AGIS. The AGIS system provides the changes and updates as well as new entries to the PanDA workload management and data management systems, which allows to reduce the human factor influence on the ADC overall efficiency. Thus WLCG and in particular ADC can be considered as an autonomic system, with implementation of all four major properties. However the autonomic level is not sufficiently high for the Tier 1 and Tier 2 computing centres, where the complete involvement of the system administrators and IT professionals in the management processes still plays a crucial role.
Development and deployment of an autonomic computing concepts for the cloud sys-

tems also gets major attention in terms of automated provisioning of the resources to the cloud platforms. The major goal of the research in terms of autonomic computing concepts integration in the cloud systems is oriented on avoiding the SLA violations, which affects the income rate the reliability reputation of the service provider. Cloud software like OpenStack [190], Hadoop based storage cloud systems [191], etc. already integrates the load balancing and failover techniques. The strategies for improvement of the existing computing resources reliable provisioning techniques are mainly based on implementation of various "Knowledge" component generation approaches. In case of the autonomic computing components of the grid computing the "Knowledge" component is mainly defined by an IT professionals and based on system administrators experience, while in case of cloud, the system knowledge acquisition techniques vary from agent-based [192] to machine learning approaches [193, 194]. However there is no generic autonomic system for SaaS, PaaS or IaaS platforms [195].

5.4. Service Response Time

Most of the autonomic computing approaches are based on architectural approaches, where the high level policies and the system related knowledge is defined by the system administrators and IT experts. This fact increases the risk of subjective treatment of entire infrastructure, hence its performance. Another problem is the lack of high level professionals who are able to follow the newly appearing platforms and systems, providing multiple services. In the digital world the large, distributed and heterogeneous platform management and administration becomes a big challenge, hence an automation is getting higher importance.
The initial point, when the complex system components needs to be automated is the identification of key parameters, which describe the behaviour of entire infrastructure and ongoing processes. The key parameters are used to identify the behavioural model of the complex and heterogeneous system, which is the starting point on the way of its management automation.
Most of the digital platforms have the service oriented architecture and the key parameter, that consumer is taking into account for the provided service assessment, is the Service Response Time (SRT). The SRT represents the speed of provided service, e.g. the web page loading speed or latency of data upload in the storage cloud. All commercial and scientific computing and storage service providers tend to decrease the SRT, thus provide higher availability and serviceability to more customers and users. Therefore SRT plays a crucial role in the service provisioning quality assessment. However, SRT also contains the information about the processes ongoing inside the service provider platform. It can be considered as a function of multiple hardware, software and communication parameters affecting the entire infrastructure performance. Thus analysing the SRT, as a key characterizing parameter for the digital service provisioning quality, has the high importance. Prediction and the service status identification based on the SRT measurements allows to avoid the SLA violations and in this way increase

the overall availability, reliability and serviceability of the large, distributed digital service provider.
The next chapter is dedicated to the description of the techniques and approaches for the service status identification and the SRT prediction based on the monitoring data including the Service Response Time Measurements.

6 Approach

One of the key aspects for the efficient deployment of Autonomic Computing concepts in the large distributed computing infrastructure is the proper implementation of the self-awareness component. Adding the self-awareness feature in to the computing facility management system, means quick problem detection and its proper root cause analysis, which requires aggregation and processing of the system related information. The information should contain the proper metrics and characteristics of dynamic processes ongoing at the computing platform. The best sources, containing and providing such an information, are the monitoring applications, which contain number of efficient tests and predefined key metrics that characterize the availability and serviceability of the entire infrastructure and the dynamics of ongoing processes. Once the monitoring information is available the next step is its analysis, which implies either involvement of experts, i.e. IT professionals and experienced IT systems administrators, or application of the data analysis methods. Since the objective for the Autonomic Computing as well as for this research is to reduce the human experts involvement in the computing infrastructure management and operations, this section will be dedicated to the overview and description of the machine learning and statistical data analysis methods used here for the digital service status identification and prediction.
Application of machine learning and statistical data analysis methods for the automation of management of complex structures and systems remains an active topic of research and covers number of industrial and scientific areas. The distributed computing platforms, e.g. computing centres involved in WLCG, have the complex structure and consist of embedded systems, thus it can be considered as a possible realm for implementation of such approaches. For efficient and proper application of the data analysis techniques it is necessary to identify the objective of these methods applications. Objectives, can be identified by modelling the activities of system administrators and IT professionals in the moment or during the process of problems detection and analysis. Activities of IT

professionals, which require analytical work are mainly related to the proper problem identification and its root cause detection as well to the anticipation of the possible service degradation or disability. The source for decisions about the current and possible states of the service are three components: the current and historical monitoring data from relatively long periods of time, experience and knowledge. These three components allows to identify the certain patterns, which can characterize current or future state of the service as well as its dynamics in time. Thus, the analytical tasks for the service state detection and prediction can be considered as a problem of pattern detection and the time series analysis.
This section is dedicated to the description of the Fuzzy Sets Theory [196] and Artificial Neural Networks [197] techniques and the approach, which is based on both techniques. The hybrid approach, described here, is actively used for the pattern detection, time series analysis, non-linear system modelling etc. Its efficiency has been proven in a number of case studies, which served for us as a motivation to apply if for the service failure identification and prediction problem. Before the description of the methods it will be provided the overview of the big picture of system management automation approaches and in its scope the online failure prediction systems.

6.1. Related Work

Implementation of self-awareness capabilities for the management automation of large and complex systems can be considered as one of the main objectives of autonomic computing and similar approaches. The key feature of such an automation systems is the capability of self-adaptation. The self-adaptation feature assumes constant monitoring and analysis of internal and external factors influencing the system and planning of the corresponding actions for avoiding system availability and serviceability degradation. The systems with self-adaptation capabilities include service degradation or failure detection and anticipation which is the main objective of this research. In this subsection, the main characteristics and approaches of the self-adapting systems will be reviewed This subsection also includes an overview of the online failure prediction tools and techniques as a large sub-class of the self-adapting systems as well as stand-alone approaches used for different type of hardware or service failure detection and forecasting.
Research in the direction of the self-adaptive systems became a hot topic especially during the last decade. There are several different definitions for self-adapting systems. According to [198] self-adaptive systems are completely autonomic, i.e. independent from the human interaction and are able to detect the changes in the environment and adjust big number of system parameters to the changes. Another conceptual definition is provided by [199], which assumes that self-adaptation is possible to implement using multi-model systems. The adaptation component in this definition assumes an existence of number of models guaranteeing the system stability in different challenging conditions, which can be applied in different challenging conditions. Despite to the provided definitions [200] claims that all self-adaptive systems are closed-loop-systems that depend on user's requirements, system properties and environmental conditions.

The [201] publication suggests a similar definition to the aforementioned ones, but with an emphasis of implementation of the self-adaptation capabilities only on the software level of the entire infrastructure, i.e. the central part of the definition is the objective for the software systems, i.e. the avoiding the violation of the QoS conditions.
Similar to the definitions there are different classifications and taxonomies of the self-adaptive systems. According to [202], which is one of the highly cited recent publications covering the taxonomy of autonomic systems, the self-adaptive approaches can be classified according to the their management workflow, hence three main groups of such systems are identified.
In [202] the self-adaptive systems with centralized and decentralized workflow are grouped as a single class. The centralized systems are managed by a single entity that aggregates internal and environmental metrics of the system and constantly assess the state of the entire infrastructure, which serves as a basis for further decisions. Examples of such systems are described in [203] and [204]. In these publication the main assumption is an efficiency and simplicity of integration of the external control mechanism for centralization of entire infrastructure management processes.
The decentralized systems do not have a single management component. Hence each of the infrastructure's sub-system is self-managed and overall performance depends on availability and serviceability of each sub-component as well as on communication among them. Such systems are also considered as approaches based on component based software engineering (CBSE) concept [205]. Systems using the CBSE concept based approach are in active usage and demand constant interaction for changing or modification of its subcomponents. The details of the concept application application are covered in [206, 207] where the structure and functionality of highly scalable and dynamic systems are described.
Another group of self-adaptive approaches are characterized as top-down and bottom-up systems [202]. The top-down systems are similar to the centralized ones and also use the centralized controller as a single management entity for assessing and guiding the performance of the entire infrastructure from top to the bottom, which is not the case for the bottom-up systems. The bottom-up approaches are identical to the decentralized adaptive systems, where the overall performance and efficiency depends on the lower, self-managed components, where the communication among them has a key role.
Systems based on decentralized or bottom-up concept are proposed in [208, 209] as a multi-agent, self-adaptive implementations. The architecture of described implementations are similar to the decentralized and bottom-up systems, i.e. it consists of multiple independent components aware about their availability and serviceability, which serves as a basic information for system's overall performance and functionality. There is no central management component and the communication has the key importance.
The feedback-latency and environmental based systems is the third class of self-adaptive approaches, which is prototyped according to the adaptation capability of the biological and social entities [198]. Ability of biological organisms to adapt themselves to the dynamically changing environmental and internal conditions, in order to survive as an individual or species, attracted researches attention during last several decades. Math-

ematical modelling of these systems was the main interest especially for the researches working in the direction of development an autonomous and stable approaches. In terms of the stable and highly reliable computing systems development the key importance is efficient integration of software and hardware components as it is described in [210]. Another direction is also the self-organization of groups of insects, which also finally guarantees the achievement of the goal. The integration of this concept is described in [211]. The feedback approach described in [212,213] applies the idea of feedback-loop, i.e. any autonomic system, which needs to analyse internal and environmental changes has to be considered as self-aware system as well, which is impossible to achieve without analysing the results of the issue actions. Therefore the key aspect goes into the loop-back component.
Due to the highly active and expanding scientific community working on the development of autonomic systems some approaches do not fit the described classification, e.g. the model-derive self-adaptive systems described in [214], [215] and [199] rely on the dynamic code generation according to the predefined abstract model. The key difficulty of this approach is development of an efficient and adequate abstract model able to cover and include all the details of the large and complex systems functionality.
Despite to the number of differences among the self-adaptive approaches the fundamental components of architecture are identical and in particular there are four main blocks Monitoring, Analysis, Planning and Executing (MAPE) (see fig.6.1).

Since the objective of this research is not the entire self-adaptive systems, but only the

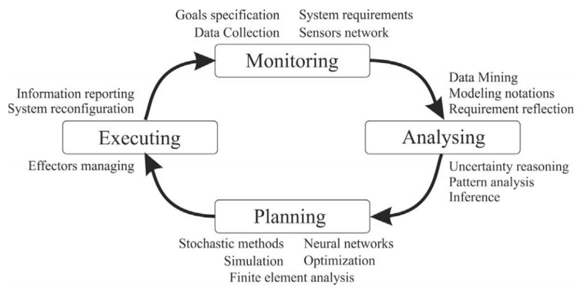


Figure 6.1.: Basic components of self-adaptive systems - MAPE [202].

failure detection and prediction approaches, the rest of the subsection will be dedicated to the overview of main approaches in this direction. However to complete the picture of the self-adaptive systems the schematic view of self-adaptive systems according to the key features is provided (see fig.6.2).

Two main types of failure detection and prediction systems can be identified, the online and offline. The online systems have higher practical importance and this research is dedicated to the provisioning and validating of such an approach.
For the overview of the failure detection and prediction systems the definition of failure itself needs to be provided. The goal of the system administrators is to provide the stable services and in this way avoid the QoS violation, therefore here the failure, i.e. degradation of the service will be defined. According to [217] there are a number of

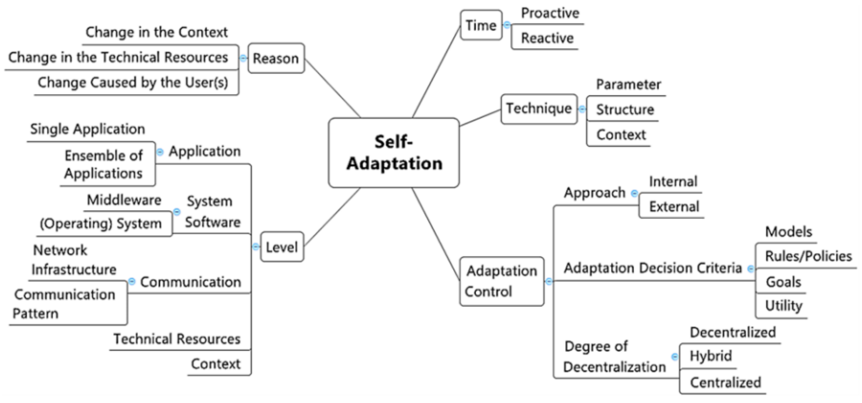


Figure 6.2.: Detailed taxonomy of self-adaptive systems [216].

definitions for the service failures and all of them can be considered in terms of QoS conditions violation. Hence any behaviour of the service, which does not meet the predefined QoS agreements can be considered as a service failure. There are number of ways for detection of the QoS violation. The common technique is to check the corresponding service log information for the keywords like: "FAILURE" or "ERROR". This way is time-consuming and does not allow to identify the service degradation quickly. Another option is to perform the service test, i.e. run the simple request for the service and if it will fail it will be assumed that QoS is violated. In this case the only information, which will be available is that service is not available. For in time fixing of the problem the failure root cause analysis needs to be performed, which again brings us to the check of the service log information. This might not lead to the failure message identification, because reason of the problem can be related to the infrastructural rather to the service. Thus proper in time identification of the service status as well as its prediction has the high importance, which is reflected in the large number of proposed systems and developed approaches. The taxonomy of basic methods used for such systems is depicted on the corresponding diagram (see. fig. 6.3).

The approach described in the scope of this research, belongs to the Fuzzy Classifier (2.2.2) group for the service failure detection and Time Series Prediction (2.4.2) class in terms of service degradation or its failure prediction. Application of fuzzy sets theory based approaches in management automation is not nova [218–220] and mostly this methods are used in combination with other approaches, like specific neural networks and genetic algorithms. The scope of application of the fuzzy sets based methods is the detection and root cause analysis of the computing system and prediction of software components ageing in the complex systems.

The time series analysis and prediction methods are mostly used for the identification of the periods of the service critical load [221–223] as well as for the forecasting of the cpu, memory and network consumption rates. The fuzzy sets and time series analysis based techniques aim to provide methods for avoiding the service failure or degradation as well as ways of the problem root cause analysis.

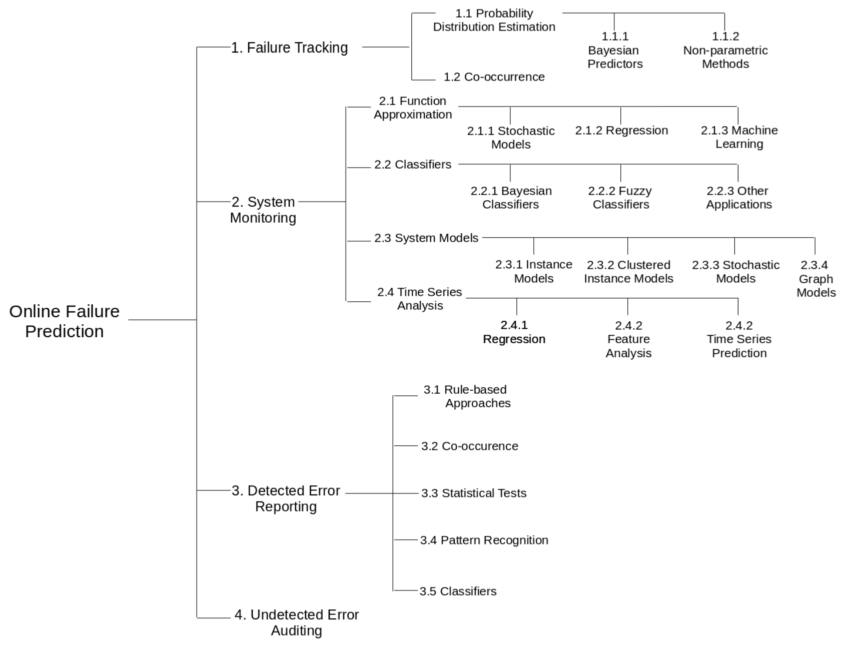


Figure 6.3.: Taxonomy of online failure prediction systems [217].

In scope of existing techniques, the application of fuzzy sets based approach for the service failure detection and its prediction by SRT can be considered as sufficient and easy to use contribution for the automation of service related problems proactive handling. The details of the approach are described in the following subsections and the results of its application are shown in the next chapter of this thesis.
Besides the activities in terms of self-adaptation and self-awareness research, there are standalone projects, providing the methods mostly for the computer hardware failure prediction. The largest share of the research in hardware problem forecasting takes the hard drive related topics [224, 225], which is also important for the general research in the direction of computing systems automation, since hardware has the same critical importance as software for the entire infrastructure.

6.2. Adaptive Network Based Fuzzy Inference System

Modelling of the digital service availability in order to identify its current state and provide its short and long term availability forecasts is the primary goal of this research. Modelling, in this case means the identification of the service failure and its availability patterns as well as studying the service behaviour in time. As a core technique for the pattern identification the Artificial Neural Networks (ann) can be considered, but these methods are not sufficiently flexible for avoiding the risks of being trapped in the local and temporal behaviour patterns. Therefore ann is usually used in conjunction

with other machine learning techniques. Hence, for sake of pattern identification generalization the Fuzzy Inference System (fis) [226] is employed. The fis allows to model the knowledge of IT professionals and system administrators, in order to perform the human like inference and decision making. Conjunction of ann and fis based approaches allows dynamic tuning of the system behaviour model with ability of general knowledge aggregation.
This subsection is dedicated to the description of the basics of the Fuzzy Sets Theory and the conceptual definitions of fuzzy variable, fuzzy rules, inference systems and conjunction of the ann and fis techniques.

6.2.1. Basic Concepts of Fuzzy Sets Theory

In everyday conversations we tend to use combination of adjectives and adverbs to characterize different objects or events. When applying such words, understanding of related crisp values is very subjective and there are different thresholds to characterize for example a height of men or women. Therefore crisp set e.g. $A = \{a|a \geqslant 190\}$cm cannot be used for classification of tall people because according to this formulation a person whose height is 189.9cm does not belong to the set A, while in colloquial language the person with such a height would be considered as tall. Thus we are facing some limitations in application of crisp sets for characterizing the metrics used in everyday life.
In 1965 L.A.Zadeh, in his publication [227] proposed the concept of fuzzy sets, i.e. the sets without crisp boundaries. The core of the concept is the idea of membership function. The membership function makes possible to map a certain crisp number to a specific characterizing terms used for a particular object or activity at a certain level.
If we assume that $\mathbb{R}$ is the real number line then elements of classical set $A \subset \mathbb{R}$ can be represented using the binomial characteristic function, which equals to 1 if $x \in A$ and 0 if $x \notin A$. In case of fuzzy sets each element is a fuzzy variable and it belongs to a particular fuzzy set with a certain level, characterized by the value of the membership function $\mu(x)$. The $\mu(x)$ can get values between 0 and 1, thus it can be considered as an extension of the characteristic function of the classical sets.
Definition 6.2.1.1: Fuzzy variable $(x, \mu(x))$ is a pair of the element of crisp value x and the membership function $\mu(x)$ describing at which degree the x belongs to the particular fuzzy set.
Definition 6.2.1.2: Membership function (MF) $\mu(x)$ of fuzzy variable $(x, \mu(x))$ characterize the membership grade of the x to a particular fuzzy set by getting the meaning from the $[0; 1]$ interval.
Definition 6.2.1.3: Fuzzy set A is the set of ordered pairs $(x, \mu(x))$, where $x \in X$, where X is the set or subset of the classical, crisp values.
Fuzzy sets, similar to classical sets, contain large number of elements, such that it is impossible and inconvenient to list all of them for the set representation. However, in case of fuzzy sets knowing of the membership grade of each fuzzy variable is crucial. Therefore, fuzzy sets are represented with the membership functions in analytical form. Here is a list of most applicable membership functions in practice.
Definition 6.2.1.4: A triangular MF has the following structure

$$triangle(x; a, b, c) = \begin{cases} 0, x \leq a; \\ \frac{x-a}{b-a}, a \leq x \leq b; \\ \frac{c-x}{x-b}, b \leq x \leq c; \\ 0, c \leq x \end{cases} \tag{6.2.1.1}$$

where the x is the element of the crisp set, while a, b and c are the parameters, with $(a < b < c)$ relationship and defining the horizontal coordinates of the triangle angles.
Definition 6.2.1.5: A trapezoid MF, with four a, b, c, d parameters and the crisp value x has the following structure

$$triangle(x; a, b, c) = \begin{cases} 0, x \leq a; \\ \frac{x-a}{b-a}, a \leq x \leq b; \\ \frac{d-x}{d-c}, c \leq x \leq d; \\ 0, d \leq x \end{cases} \tag{6.2.1.2}$$

The parameters are in the following relationship $(a < b < c < d)$ and indicate the horizontal coordinates of the trapezoidal MF angles.
The triangular and trapezoidal MFs are actively used for the real-time modelling due to their simple structure, however for modelling of the complex, non-linear systems they are not efficient due to their non-smooth structure, i.e. non-differentiability. Another group of membership functions, does not have the sharp edges and allows the smooth transition from one fuzzy characteristics to another. Also their differentiability makes them applicable for modelling of the dynamic and complex structures as well as activities.
Definition 6.2.1.6: Gaussian MF depends on two parameters c, the centre of the curve and σ, the width and have the following structure

$$gaussian(x; c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{6.2.1.3}$$

Definition 6.2.1.7: Generalized bell MF depends on three parameters a, b, c in the following way

$$bell(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}} \tag{6.2.1.4}$$

The parameter a regulates the steepness of the curve, the sign of b defines the vertical position of the curve as well as its width and the parameter c identifies the centre of the curve.
The Generalized bell shaped MF is also referred as the Cauchy MF, since it is related to the Cauchy distribution applicable in the probability theory.
The Gauss and Generalized bell MF are very efficient and precise for the description of the complex processes, but the Gauss MF has advantage in terms of the computing, since it contains less free parameters that needs to be calculated and shows adequate efficiency in a number of applications.
Definition 6.2.1.8: The Sigmoid MF depends on two parameters the a and c, where a regulates the slope of the curve when $x = c$

$$sig(x; a, c) = \frac{1}{1 + exp[-a(x - c)]} \tag{6.2.1.5}$$

The Sigmoid MF is efficient for applying in cases of the extreme conditions, i.e. when the modelling requires to take into account terms like "very small" or "too positive", which is required not frequently and is out of scope of our interest.

Fuzzy sets are similar to the classical sets, hence the set operations like intersection

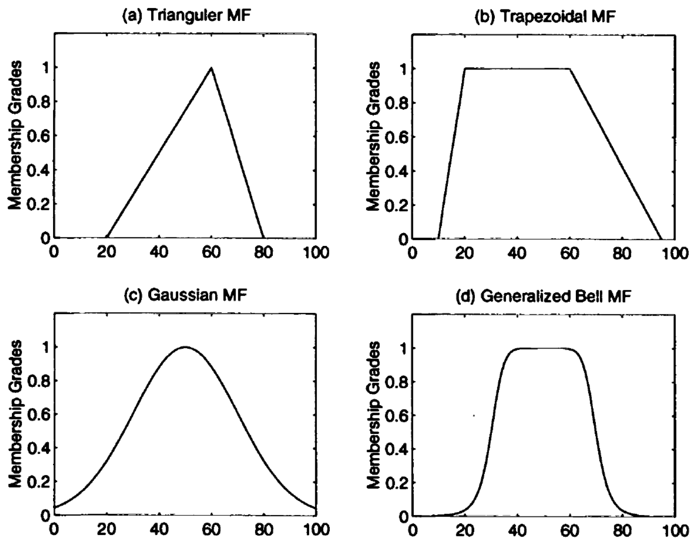


Figure 6.4.: Examples of Four Main Membership Functions [228].

and union also exist. For specification of two, A and B fuzzy sets intersection the $T : [0, 1]\text{X}[0, 1] \rightarrow [0, 1]$ is employed, which allows to aggregate corresponding membership grades in the following way:

$$\mu_{A \bigcap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \widetilde{*} \mu_B(x) \tag{6.2.1.6}$$

Here $\widetilde{*}$ represents the class of fuzzy intersection operators, which are also named as $T - norm$ (triangular norm) operators [229].

Definition 6.2.1.9: $T - norm$ is a function of two arguments satisfying the following conditions:

$$\begin{aligned} T(0,0) = 0, T(a,1) = T(1,a) = a, &\quad \text{(boundary)} \\ T(a,b) \leq T(c,d), \quad \text{if} \quad a \leq c \quad \text{and} \quad b \leq d &\quad \text{(monotonicity)} \\ T(a,b) = T(b,a) &\quad \text{(commutativity)} \\ T(a, T(b,c)) = T(T(a,b),c) &\quad \text{(associativity)} \end{aligned} \tag{6.2.1.7}$$

There are classes of functions, which satisfy the $T - norm$ conditions, but in practice the following four functions are applied actively.

$$\begin{aligned} T_{min}(a,b) = min(a,b) = a \wedge b, &\quad \text{Minimum} \\ T_{ap}(a,b) = ab &\quad \text{Algebraic product} \\ T_{bp}(a,b) = 0 \vee (a+b-1) &\quad \text{Bounded product} \\ T_{dp}(a,b) = \begin{cases} a, & \text{if} \quad b = 1; \\ b, & \text{if} \quad a = 1; \\ 0, & \text{if} \quad a,b < 1 \end{cases} &\quad \text{Drastic product} \end{aligned} \tag{6.2.1.8}$$

Similarly to the fuzzy intersection, the fuzzy union operation can be represented as $S : [0,1] \text{X} [0,1] \rightarrow [0,1]$ or

$$\mu_{A \bigcap B}(x) = S(\mu_A(x), \mu_B(x)) = \mu_A(x) \widetilde{+} \mu_B(x) \tag{6.2.1.9}$$

where, $\widetilde{+}$ represents the class of operators satisfying the $T - conorm$ [229] conditions.

Definition 6.2.1.10: $T - conorm$ or $S - norm$ can be considered as a two place function satisfying following conditions:

$$\begin{aligned} S(1,1) = 1, S(a,0) = S(0,a) = a, &\quad \text{(boundary)} \\ S(a,b) \leq S(c,d), \quad \text{if} \quad a \leq c \quad \text{and} \quad b \leq d &\quad \text{(monotonicity)} \\ S(a,b) = S(b,a) &\quad \text{(commutativity)} \\ S(a, S(b,c)) = S(S(a,b),c) &\quad \text{(associativity)} \end{aligned} \tag{6.2.1.10}$$

Similarly to the $T-norm$ operators, large number of functions meets the requirements shown in the definition (6.1.2.10) but the following four functions are actively used in practice.

$$
\begin{aligned}
S_{max}(a,b) = max(a,b) = a \vee b, &\quad \text{Maximum} \\
S_{as}(a,b) = a + b - ab &\quad \text{Algebraic sum} \\
S_{bs}(a,b) = 0 \wedge (a+b) &\quad \text{Bounded sum} \\
S_{ds}(a,b) = \begin{cases} a, & \text{if} \quad b = 0; \\ b, & \text{if} \quad a = 0; \\ 0, & \text{if} \quad a,b > 0 \end{cases} &\quad \text{Drastic sum}
\end{aligned}
\tag{6.2.1.11}
$$

Besides the union and intersection the fuzzy complement [229] is also considered as one of the key fuzzy sets operations.
Definition 6.2.1.11: The fuzzy complement is a continuous function satisfying the following conditions:

$$
\begin{aligned}
N(0) = 1 \quad \text{and} \quad N(1) = 0, &\quad \text{(boundary)} \\
N(a) \geq N(b), \quad \text{if} \quad a \leq b &\quad \text{(monotonicity)} \\
N(N(a)) = a &\quad \text{(involution)}
\end{aligned}
\tag{6.2.1.12}
$$

According to the literature the Sugeno's [230] and Yager's [231] fuzzy complements are used in most of the practical cases 6.2.1.13.

$$
\begin{aligned}
N_s(a) = \frac{1-a}{1+sa} &\quad \text{Sugeno's complement} \\
N_\omega(a) = (1-a^\omega)^{\frac{1}{\omega}} &\quad \text{Yager's complement}
\end{aligned}
\tag{6.2.1.13}
$$

As it is shown in the formulas 6.2.1.13 Sugeno's and Yager's complement functions depend on a single parameter. For Sugeno it is $s > -1$ and for Yager's it is a positive ω. Each particular value of s and ω allows to identify the individual complement function. Linguistic variable is a fundamental concept of fuzzy sets theory and has the same importance as the membership functions. Examples of linguistic variables are "height", "age", etc. Instead of crisp values the variables are taking the meaning of terms, e.g. "young","old","small","tall", etc. These terms are called the linguistic values of the linguistic variable. Clearly there is a connection between the numerical representations of age and height and the linguistic terms, which is possible via the membership functions. For each of the linguistic value the corresponding membership function is defined, with its individual set of the parameters (see fig. 6.5). The number of possible linguistic

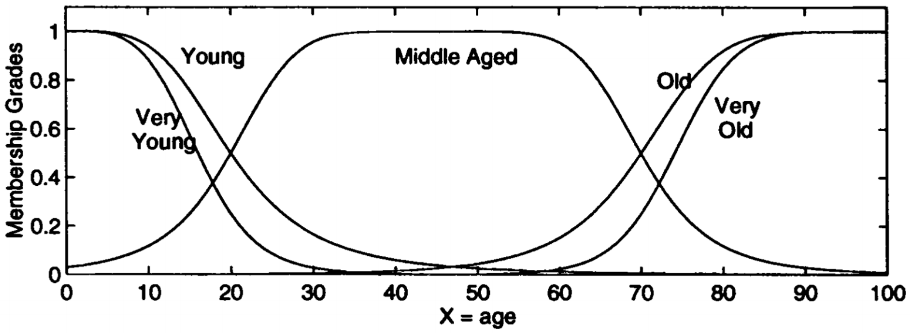


Figure 6.5.: Examples of Linguistic Values and Corresponding MFs [228].

values for a particular variable can be defined with the help of experts or automatically.
Existence of linguistic terms allows to compose the linguistic structures with the certain logic behind it, as for human beings when composing the sentence. However, despite to the sentence used in the colloquial speech, the structure of the "sentences" composed from the linguistic variables have the form of $if - then$ rules (see 6.2.1.14), where A and B represent the linguistic values. The $x \quad is \quad A$ part of the rule is called the antecedent or premise, while $y \quad is \quad B$ is the consequence or conclusion part of the rule.

$$if \quad x \quad is \quad A \quad then \quad y \quad is \quad B \tag{6.2.1.14}$$

These rules also represent a fundamental concept in the fuzzy sets theory and makes possible to apply it in the fields of the system identification and process automation, time series prediction etc. The premise part of the rule usually have more complex structure and contains a number of linguistic values connected with logical operations like *and* and *or* (see 6.2.1.15). The logical operations are identical to the three main operations the union, intersection and complement and can be represented as 6.2.1.16.

$$if \quad x_1 \quad is \quad A \quad and \quad x_2 \quad is \quad B \quad or \quad x_3 \quad is \quad C \quad then \quad y \quad is \quad D \tag{6.2.1.15}$$

$$\begin{aligned} \mu_{A\cap B} = min(\mu_A(x), \mu_B(x)) = \mu_A(x) \times \mu_B(x), \quad "and" \\ \mu_{A\cup B} = max(\mu_A(x), \mu_B(x)) = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x) \quad "or" \\ \mu_{\overline{A}}(x) = 1 - \mu_A(x) \quad complement \end{aligned} \tag{6.2.1.16}$$

Definition of logical operations that can be used for connecting the multiple parts of the premise and consequent parts of the rules makes possible to get the single output

from a particular fuzzy rule, which needs to be defuzzified [228]. The defuzzification methods allow to extract the crisp values from the linguistic value, or corresponding fuzzy variable, which makes the derived results applicable for other systems based on the approaches using the concepts of classical sets. In the literature, the number of defuzzification methods are described, like method of center of gravity (COG), mean of maxima (MOM), bisector of area (BOA) [232], which can be considered as equally efficient in most of the cases.
Fuzzy sets operations, linguistic variables and values and fuzzy rules makes basis for the practical application of the fuzzy sets theory concepts. Example of the general method, which is based on the fuzzy approach and proved its efficiency in many case studies is the fuzzy inference system, which will be described in the following sub-section.

6.2.2. Fuzzy Inference System

The fuzzy inference system (FIS) allows to apply the capabilities of fuzzy sets theory in practice. The inference system is a computing framework, which takes the crisp or fuzzy values as an input and produces a single output. The final output in most of the cases of the fuzzy inference systems is a crisp value, which is derived using one of the defuzzification methods [196]. Besides the defuzzification component, the inference system consists of fuzzification, fuzzy rule base, set of membership functions and fuzzy inference engine components (see fig. 6.6).
The fuzzification and MF set components allow to transform the crisp or fuzzy variables

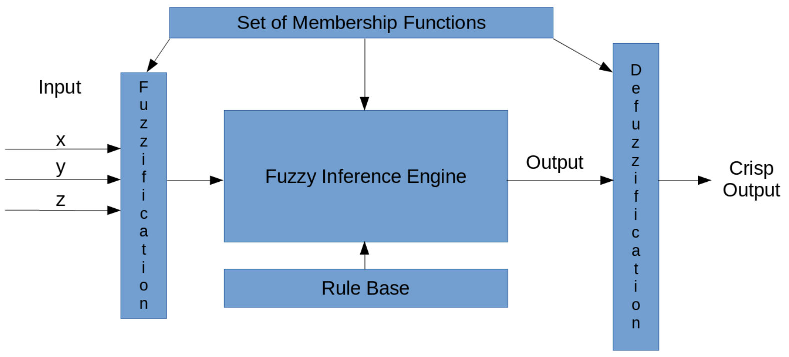


Figure 6.6.: The Fuzzy Inference System.

into the linguistic values with the corresponding MF degrees. The rule base component consists of number of fuzzy inference rules with "and" or "or" logical connections. The fuzzy inference engine is the core component of FIS and defines how the fuzzy if-then rules can be converged into the fuzzy output, which finally should be transformed into the crisp value using one of the defuzzification methods.
According to the literature there are three main defuzzification methods used in practice. The Centroid of Area COA 6.2.2.1, Bisector of Area BOA 6.2.2.2 and Mean of Maxima

MOM 6.2.2.3 [228] (see fig. 6.7).

$$COA = \frac{\int_Z \mu_A(z) z dz}{\int_Z \mu_A(z) dz} \tag{6.2.2.1}$$

$$\int_{\alpha}^{zBOA} \mu_A(z) dz = \int_{zBOA}^{\beta} \mu_A(z) dz \tag{6.2.2.2}$$

$$MOM = \frac{\int_{Z'} z dz}{\int_{Z'} dz} \tag{6.2.2.3}$$

All the aforementioned defuzzification methods are meant for derivation of the crisp value from the fuzzy set A defined on the Z classical set of real numbers. In case of COA, the $\mu_A(z)$ is the cumulative output MF. The COA is the most widely used method for transforming the fuzzy values into crisp. For BOA $\alpha = min\{z|z \in Z\}$ and $\beta = max\{z|z \in Z\}$ and for MOM, the $Z' = \{z|\mu_A(z) = \mu^*\}$, where μ^* is the maximum value of MF, which can be reached by maximizing the z, i.e. the input crisp value.

Based on the types of fuzzy inference engine and rules there are distinguished three

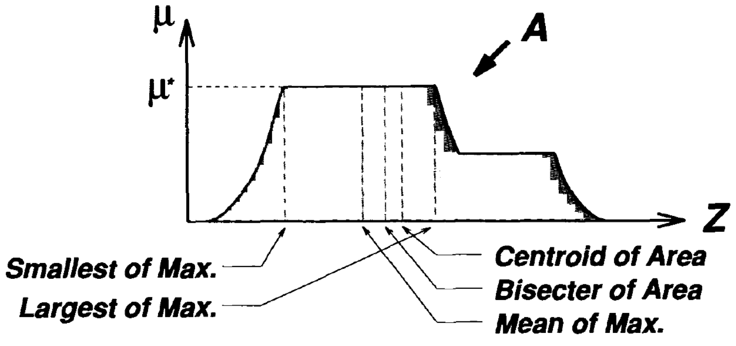


Figure 6.7.: Graphical depiction of defuzzification methods [228].

main modifications of fuzzy inference systems.

The Mamdani [233] FIS (see fig. 6.8) represents the group of systems, where the final output of the system is defined using the "max" 6.2.1.16 operation to the outputs of the fuzzy rules. The output of each fuzzy rule is the minimum of the rule firing strength and the consequent membership function.

In case of the Tsukamoto [234] type of inference systems (see 6.9) the output is weighted average of each rule's output. The output of the rule in case of these systems is induced by the rule's firing strength, hence it is a crisp value. The only requirement for the

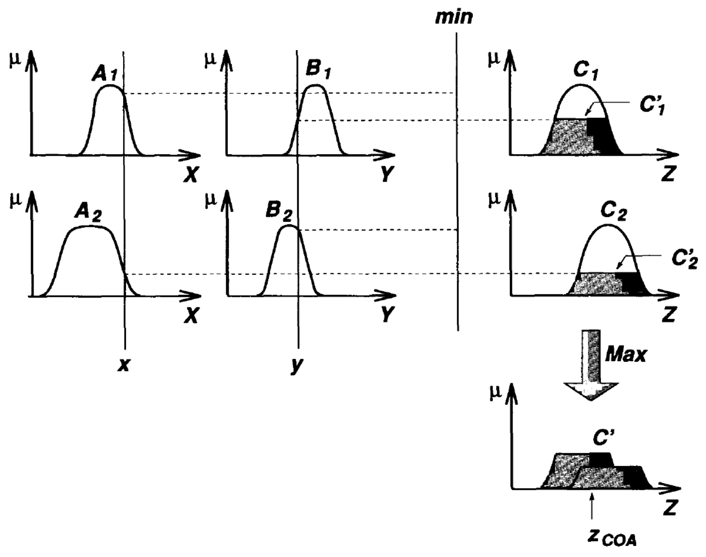


Figure 6.8.: Mamdani fuzzy inference system [228].

Tsukamoto FIS is the monotonicity of the MF of the fuzzy set, which represents the rule's consequent part.

The third group of fuzzy inference system is called after Takagi, Sugeno and Kang [235, 236] or TSK. The output of the TSK system is the weighted average of each rule as in case of Tsukamoto systems. However, in this case the output of each rule is the linear combination of crisp inputs and constant (see fig. 6.10). If the linear combination is the first order polynomial the system is called as first-order Sugeno fuzzy model and if the rule's output is constant then the FIS is called the zero-order Sugeno model.

According to the structure of groups of FIS, the defuzzification is required only for the Mamdani inference system, which makes TSK and Tsukamoto models most popular in practice.

6.2.3. Feed Forward Artificial Neural Network

Most of the problems that require machine learning and statistical data analysis methods for their solution can be grouped into the classification and regression types of problems. Solution of classification problem requires identification of the mapping of single or multi-dimensional space to the finite number of discrete values. In case of regression problem, finding the solution means determination of the adequate mapping of the input space to the continuous output variable, where the output variable can be a single or multidimensional value.

Application of classical methods, based on linear combination of basis functions [237], for solution of the classification and regression problems are usually facing curse of dimensionality issues.

The Artificial Neural Networks (ANN) based approach is an efficient way for avoiding

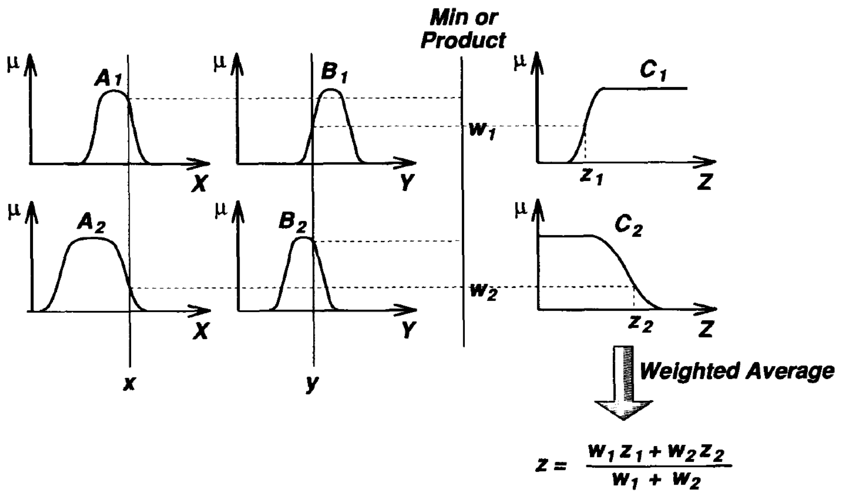


Figure 6.9.: Tsukamoto fuzzy inference system [228].

the problems related to the large dimensions. The development of ANN based methodology was inspired in the process of modelling the information processing capabilities of the biological organisms [238]. Nowadays it is impossible to imagine the field of machine learning without ANN based approaches, which are actively used in the dynamic modelling of complex processes and systems, for pattern recognition etc.
Each process or the system under the observation has the set of characteristic variables $\{x_n\}$ and $\{t_n\}$, where $n = \overline{1,N}$. These variables are in certain relationship between each other, usually this is a binary relationship, $F : x_n \to t_n$. The goal of the methods used for the solution of classification and regression problems, as well as ANN based approaches is to approximate these relationships with $y(x_n)$, i.e. provide the approximate value of t_n, when x_n is given as an argument, i.e. $y(x_n) = t_n$, hence $y \sim F$.
The simplest solution of this problem is known as a linear regression model [237] (see 6.2.3.1), where $X_k = (x_1, ..., x_k)$ is the certain subset of $\{x_n\}$ and $\Omega_k = (\omega_1, ..., \omega_k)$ is the set of the corresponding parameters, which needs to be calculated. However the linear model is not capable to adequately deal with the complex processes and systems, therefore as an extension of an approach the linear combination of non-linear functions, of input variable elements is taken, 6.2.3.2 instead of linear combination of the variable elements itself. In 6.2.3.2 $\phi_i(x)$ are called the basis functions where $\phi_0(x) = 1$. The basis functions need to be defined in advance and there is a set of actively used examples in practice [239].

$$y(X, \Omega) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + ... + \omega_k x_k \tag{6.2.3.1}$$

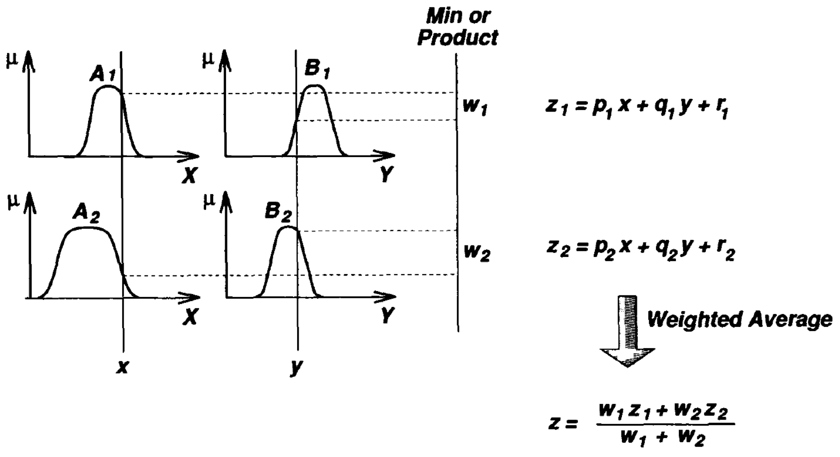


Figure 6.10.: TSK or Sugeno fuzzy inference system [228].

$$y(X,\Omega) = f(\sum_{i=0}^{k} \omega_i \phi_i(x)) \tag{6.2.3.2}$$

In the described regression method it is also assumed that the number of basis functions is undefined, which complicates the practical application of this method in terms of parameter calculation. Especially this issue is getting bigger importance, when the input variable dimension is high, e.g. order of several hundreds.
The ANN approach suggests to define the number of basis functions in advance and to keep it small, up to ten, which of course leads to the loss of the approach accuracy. To compensate this disadvantage it is suggested to use the parametrized form of the basis functions, which will be adjusted according the available input X_k and output $T_p = \{t_1, t_2, ..., t_p\}$ datasets and relationships among them.
Let us assume that we have $k+1$ dimensional input vectors $\{x\}_{i=0}^{k+1}$, where $x_0 = 1$ and p dimensional output vectors $\{t_j\}_{j=1}^{p}$. The simplest approach, according to the ANN methodology, to provide the parametrized $\{\phi_j\}_{j=0}^{M}$ bases functions is the consider it as a linear combination of the input variables (see 6.2.3.3), where $x_0 = 1$ and $j = \{1, ..., M\}$. The (1) in the 6.2.3.3 indicates that $\omega_{ji}^{(1)}$ parameters belong to the first layer of the ANN and they are refered as weights. The Ψ_j are so called activation quantities which are transformed by the differentiable, non-linear activation function $h()$ (see 6.2.3.4).

$$\Psi_j = \sum_{i=0}^{k} \omega_{ji}^{(1)} x_i \tag{6.2.3.3}$$

$$Z_j = h(\Psi_j) \tag{6.2.3.4}$$

The Z_j corresponds to the outputs of the 6.2.3.2 named as hidden units. In order to map the elements of the input variables to the components of the output the 6.2.3.4 quantities are once again linearly combined (see 6.2.3.5).

$$\Xi_p = \sum_{j=0}^{M} \omega_{pj}^{(2)} Z_j, p = \overline{1, P} \tag{6.2.3.5}$$

The linear combinations produce the second layer of the network, with the $\omega_{pj}^{(2)}$ parameters, which needs to be calculated. At the final step, the last activation quantities are transformed using the activation function of choice. The only requirements to the activation function is its differentiability, so there is a vast class of functions that can act as an activation one, however according to the literature [237], there are three major functions 6.2.3.6 used widely in practice.

$$\begin{aligned} f(x) = x, &\quad \textit{linear} \\ f(x) = \frac{1}{1+\exp^{-x}} &\quad \textit{sigmoid} \\ f(x) = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} + \exp^{-x}} &\quad \textit{tangent} \end{aligned} \tag{6.2.3.6}$$

In case of combining all steps of the feed forward ANN [237] analytical form of entire approach is the following 6.2.3.7, where X is the set of input vectors and Ω is the vector of all parameters.

$$y_p(X, \Omega) = f(\sum_{j=0}^{M} \omega_{pj}^{(2)} h(\sum_{i=0}^{k} \omega_{ji}^{(1)} x_i)) \tag{6.2.3.7}$$

In order to achieve highly accurate approximation of $\{t_n\}$ with $y(X, \Omega)$ it is necessary to tune the set of parameters Ω, which is possible by minimization of the following "error" function 6.2.3.8, where N is the number of elements from X input variables, which are dedicated for the system identification, i.e. training.

$$E(\Omega) = \frac{1}{2} \sum_{i=1}^{N} \|y(X_i, \Omega) - t_i\|^2 \tag{6.2.3.8}$$

Equation 6.2.3.8 can be transformed into the following form 6.2.3.9 [237], which is a well known optimization problem.

$$\bigtriangledown E(\Omega) = 0 \tag{6.2.3.9}$$

Number of efficient numerical methods for the solution of 6.2.3.9 exist, most of them rely on initial identification of the first set of $\Omega^{(0)}$ parameters and next set of parameters are identified iteratively by moving through the parameter space 6.2.3.10.

$$\Omega^{(\tau+1)} = \Omega^{(\tau)} + \triangle\Omega^{\tau} \tag{6.2.3.10}$$

In 6.2.3.10 the τ indicates the iteration label, while the $\triangle\Omega^{\tau}$ is the update of the parameter vector. One of the most widely used methods for the iterative search of the parameter vector is the gradient descent method [240], which has the following analytical form 6.2.3.11, where the $\eta > 0$ parameter is called a learning rate and each iteration of the method means the parameter vector update in the direction of negative gradient [237].

$$\Omega^{(\tau+1)} = \Omega^{(\tau)} - \eta \bigtriangledown E(\Omega^{\tau}) \tag{6.2.3.11}$$

The integration of the gradient descent method in the ANN parameter update process is called the learning process with error backpropagation [237] algorithm and is widely adopted in a number of modifications of the feed-forward ANN. One of such modifications will be described in the following sub-section.

6.2.4. Adaptive Network Based Fuzzy Inference System

Advantage of FIS is its ability to process and extract the quantitative and qualitative aspects of data and apply the human like reasoning for modelling of the process or the system under the observation. The main disadvantage of fuzzy inference systems is its dependency on the expert, i.e. subject knowledge. For deployment of the adequate FIS, which is able to model the particular process or the system it is required to define in advance the corresponding membership functions, their parameters and the inference rules, which is impossible without involvement of experts, i.e. participation of the human factor. The experts contribution to the FIS deployment can lead to the subjective identification of the model, which can be harmful on the way of search of the problem's general solution. Despite to the FIS, the ANN based methods are able to model the vast variety of the complex systems and processes without involvement of the experts. Using the learning algorithms the ANN methods are capable to automatically define the optimal values of the parameters, in order to achieve the adequate generalization of the system or process model. However, the ANN based approach lack the features of FIS, i.e. extracting of the human-like reasoning and qualitative aspects from the information. Combination of these two methods was initially proposed by Jang [228] under the general name Neuro-Fuzzy systems. An example of widely used and applied Neuro-Fuzzy approach is the Adaptive Network-based Fuzzy Inference System (ANFIS) (see fig. 6.12).

ANFIS is five layered feed-forward neural network, which implements the Sugeno

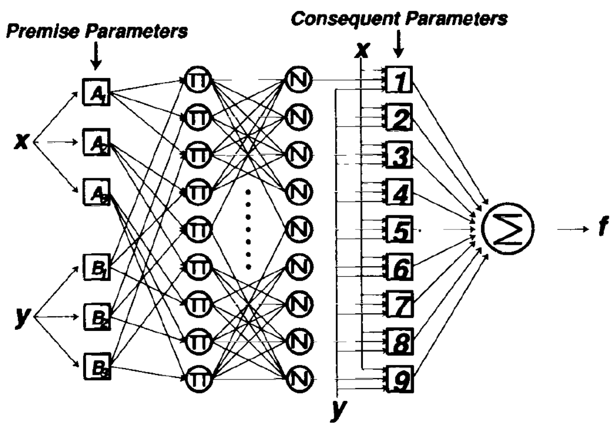


Figure 6.11.: Example of ANFIS [228].

(TSK) type of FIS. ANFIS training consists of forward and backward steps, which will be described in details below, but first it will be discussed ANFIS architecture in details. In order to simplify the description of ANFIS architecture, assume that there is an ANFIS with two input variables and two linguistic values for each ??. The first layer of ANFIS performs fuzzification, i.e. calculation of membership function values for inputs x and y (see 6.2.3.3), where $\mu_i(x)$ is the Gaussian MF and σ_i, c_i are called the premise parameters. For $i = 1, 2$ the $\mu_i(x)$ is calculated for the x, while for $i = 3, 4$ the $\mu_i(x)$ in 6.2.4.1 corresponds to the linguistic values for y.

$$O_{1,i} = \mu_i(x) = \exp(-\frac{1}{2}(\frac{x - c_i}{\sigma_i})^2), i = 1, 2, 3, 4 \tag{6.2.4.1}$$

The second layer 6.2.4.2 uses one of the fuzzy "and" operations, which meets the $T - norm$ requirements. In this case it is a multiplication 6.2.1.8.

$$O_{2,i} = \omega_i = \mu_{A_i}(x)\mu_{B_i}(y), i = 1, 2 \tag{6.2.4.2}$$

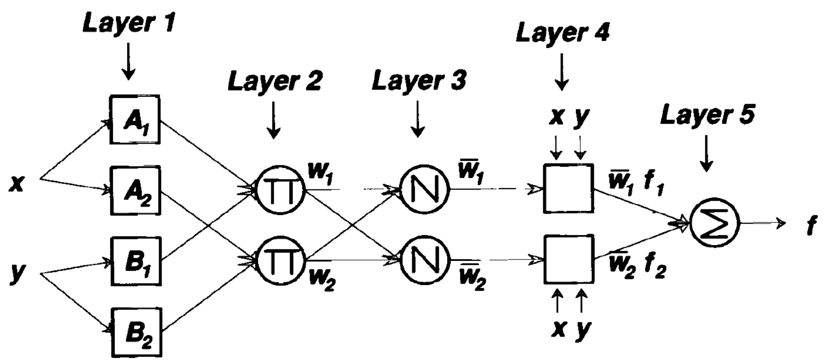


Figure 6.12.: Example of ANFIS with two inputs [228].

The following layer averages the inference rule firing strength 6.2.4.3.

$$O_{3,i} = \bar{\omega_i} = \frac{\omega_i}{\omega_1 + \omega_2}, i = 1, 2 \tag{6.2.4.3}$$

Each node of the fourth layer represents the adaptive node corresponding to the Sugeno model 6.2.4.4, where p_i, q_i and r_i are called the consequent parameters.

$$O_{4,i} = \bar{\omega_i} f_i = \bar{\omega_i}(p_i x + q_i y + r_i) \tag{6.2.4.4}$$

The final layer of the ANFIS generates a single output 6.2.4.5.

$$O_{output} = \sum_i \bar{\omega_i} f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \tag{6.2.4.5}$$

Thus, the five layer feed-forward neural network represents the Sugeno type FIS with the learning ability, hence below will be described its training procedure.
According to the ANFIS architecture, there are two types of parameters, the premise and consequent that needs to be adapted according the training data-set.
The first step of the ANFIS training process is called the forward step and it is assumed that the premise parameters are fixed, i.e. parameters equal to certain predefined values. For the calculation of the consequent parameters the Least Squared Estimate (LSE) [228] method is applied. In particular, if we consider the output of the layer three as $O_3 = F(\bar{I}, S)$, where $\bar{I}$ is the input vector and S is the set of the premise and consequent parameters. Under the assumption that we are able to apply the H function,

such that $H \circ F$ is linear to the consequent parameters the LSE can be used for the consequent parameters calculation.
If the H function exists the solution of equation $H \circ O_3 = H \circ F(\bar{I}, S)$ can be transformed as the minimization of the following problem $\|AX - B\|^2$, where A is the set of outputs from the layer 3, B is the target output and X is the set of undefined consequent parameters, i.e. subset of S. The minimization of $\|AX - B\|^2$ can be solved against X in the following way 6.2.4.6, where A^T is the transpose matrix of A and $(A^T A)^{-1} A^T$ is the pseudo-inverse of A under the condition that $(A^T A)$ is non=singular.

$$X^* = (A^T A)^{-1} A^T B \tag{6.2.4.6}$$

In order to avoid the complexities for the solution of the 6.2.4.6 equation, i.e. for handling the singularity of $(A^T A)$ the recursive LSE method was suggested by Jang [228], implemented in the following form 6.2.4.7, where $X_0 = \bar{0}$, $S_0 = \gamma I, \gamma >> 0$ and I is the identity matrix, a_i^T is the i-th row of the matrix A, b^T is the i-th element of the matrix B and $X^* = X_k$, where k is the number of the training datasets.
Once the consequent parameters are calculated the network output can be obtained, where the error measure of the k-th entry will have the following form 6.2.4.8, where the T_k is the desired, i.e. actual output and the O_k is the ANFIS output. Thus, the Root Mean Squared Error (RMSE) for the k-th training dataset have the following form 6.2.4.9.

$$\begin{cases} S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \\ X_{i+1} = X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \end{cases} \qquad \text{i=1,...,k} \tag{6.2.4.7}$$

$$E_k = (T_k - O_k)^2 \tag{6.2.4.8}$$

$$E = \sqrt{\frac{\sum E_k}{k}} \tag{6.2.4.9}$$

The backward path coincides with the typical ANN backpropagation algorithm using the gradient descent (GD) method, hence the 6.2.4.8 will be transformed as follows 6.2.4.10. On another hand, according the chain rule the error for the i-th node of the l=th layer, will have the following form 6.2.4.11. Thus the error derivative against the premise parameters S_1, which needs to be identified at the backpropagation step, will have the following form 6.2.4.12.

$$\frac{\partial E_k}{\partial O_{i,k}} = -2(T_{i,k} - O_{i,k}) \tag{6.2.4.10}$$

$$\frac{\partial E_k}{\partial O_{i,k}} = \sum_{m=1}^{N} \frac{\partial E_k}{\partial O_{m,k}} \frac{\partial O_{m,k}}{\partial O_{i,k}} \tag{6.2.4.11}$$

$$\frac{\partial E_k}{\partial S_1} = \sum_{O \in S} \frac{\partial E_k}{\partial O} \frac{\partial O}{\partial S_1} \tag{6.2.4.12}$$

Under the consideration of ANFIS structure and the fact that O represents the outputs of all the layers the 6.2.4.12 can be transformed in the following way , where the $\frac{\partial O_5}{\partial O_4} = \frac{\partial(\sum f_i \bar{\omega} i)}{\partial(f_i \bar{\omega}_i)} = 1$, $\frac{\partial O_4}{\partial O_3} = \frac{\partial(f_i \bar{\omega}_i)}{\partial(\bar{\omega}_i)} = f_i$, $\frac{\partial O_3}{\partial O_2} = \frac{\partial}{\partial \omega_i} \frac{\omega_i}{\sum_{j=1}^{m} \omega_j} = \frac{\sum_{j=1}^{n} \omega_j - \omega_i}{(\sum_{j=1}^{n} \omega_j)^2}$ and $\frac{\partial O_2}{\partial O_1} = \frac{\partial}{\partial A_m} \prod_{A_j \in set(A_m), A_j \neq A_m} A_j$, where $\bar{\omega_i}$ is the average firing strength of the i-th rule from layer four to layer five, i.e. output. The f_i is the i-th rule's consequent parameter function, n is the total number of rules in the j-th layer and $A_j \in set(A_m)$ is the membership grades for the premise part of rules, which contain the fuzzy set A_m.

$$\frac{\partial E_k}{\partial S_1} = \frac{\partial E}{\partial O_5} \frac{\partial O_5}{\partial O_4} \frac{\partial O_4}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial O_1} \frac{\partial O_1}{\partial S_1} \tag{6.2.4.13}$$

Since for the practical application, the membership function of choice is the Gaussian 6.2.1.3 the update for its σ parameter will have the following form 6.2.4.14, where L is the training epoch, i.e. the iteration label and η is the learning rate, which can be viewed as 6.2.4.15, where k is the length of the gradient transition step in the parameter space.

$$\sigma_{L+1} = \sigma_L - \eta \frac{\partial E}{\partial \sigma} \tag{6.2.4.14}$$

$$\eta = \frac{k}{\sqrt{\sum_{S_1} (\frac{\partial E}{\partial S_1})^2}} \tag{6.2.4.15}$$

6.3. Support Vector Machine

The Support Vector Machines (SVM) approaches [241] are efficiently used for the solution of the classification problems. It leads to the identification of the hyperplane that separates two classes of data with maximum margin. In particular, assume that there is a set of data pairs $(x_i, y_i), i = 1, 2, ..., l$, where $x_i \in \mathbb{R}^n$ and $y_i = \{-1, 1\}$, the SVM algorithms assume the solution of 6.3.0.16 optimization problem, under the condition $y_i(\omega^T \phi(x_i) + b) \geq 1 - \xi_i$.

$$min_{\omega,b}(\frac{1}{2}\omega^T \omega + C \sum_{j=1}^{l} \xi_i) \tag{6.3.0.16}$$

The ξ_i is the loss function with the arguments $\xi(\omega, b, x_i, y_i)$ and $C \geq 0$ is the penalty quantity of the training error. The ϕ can be identified from the kernel function $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_J)$, where the kernels may have the following analytical form 6.3.0.17, where γ, r and d are the kernel parameters.

$$\begin{cases} K(x_i, x_j) = x_i^T x_j, \quad linearkernel \\ K(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0, \quad RadialBasisFunction(RBF) \\ K(x_i, x_j) = tanh(\gamma x_i^T x_j + r) \end{cases} \tag{6.3.0.17}$$

The kernels specified in 6.3.0.17 are actively used in practice, however there are number of other kernels that can be used for other specific problems.
The SVM can be used not only for the classification but also for the feature ranking problem as well. The feature ranking problem gets higher importance, when $x_i \in \mathbb{R}^n$ has high dimension and it is required to deal only with the fewer values for avoiding the computational complexities. The feature ranking allows to identify the weights for x_i components and rank them by importance of their influence of being labelled as -1 or 1. However, only the linear kernel based SVM allows to use the $\omega \in \mathbb{R}^n$ [242] as the weight parameters for the x components importance identification.
Described methods based on SVM and ANFIS are used as the fundamental approaches for the case studies and derived results provided in the next section of the research.

7 Results

Rapidly growing demand on the digital services has motivated a number of developments and conceptual changes in the world of information technologies. The QoS requirements has dramatically changed during last decade and even short term disability of such services as communication, scientific computing, social networks, cloud storage or online shopping can affect the life of large number of people. Another motivator of rapid evolution and changes in the digital world is the increasing number and scale of the security threats.
The high demand and expectations helped to significantly improve the stability of the digital service providers. This was achieved, mostly by implementation of Service Oriented Architecture approach and number of backup, failover and virtualization technologies. These new implementations as well as the rapidly growing demand on the digital services, caused the growth of the underlying infrastructure and complicated the process of its management, administration and failure handling. These complications caused changes of the paradigm of the systems administration, monitoring and management, hence motivated to create the new tools and applications with higher level of automation, flexibility and accuracy. While the monitoring tools, which were showing and analysing the system state at the given moment of time, new monitoring tools are able to take into account all available information and identify the system's model and behavioural patterns. Such kind of new systems are still in a very mature stage of their development and are attracting more and more attention from the scientific communities. The new capabilities of the monitoring tools as well as the changes in the infrastructure and technologies opened new opportunities in the direction of system management and administration automation of the large scale computing platforms. Hence, as a result the concept of the Autonomic Computing has evolved.
One of the key capabilities of Autonomic Computing is the proactive management, i.e. decreasing the number of system or service failures by in advance knowledge about the

risks. The core concept of the proactive management is analysing of the available information about the systems and services. Thus, the most important constituent of such tools are the machine learning and statistical data analysis algorithms. Therefore, it is important to show the efficiency of a particular data analysis approach before its integration in a particular monitoring or management tool.
This chapter describes the monitoring and system's related data analysis approaches based on ANFIS and SVM methodologies. The first approach is able to identify the failure and perform its root cause analysis, while the second does the service degradation and failure prediction. Results are described in the scope of case studies and are presented in details in the following subsections.

7.1. The Service Status Identification

The QoS measurement of the provided services makes possible to assess the efficiency, reliability and availability of the digital service provider infrastructure. High QoS and minimized occurrences of the SLA violations, among the service consumers and providers, can be guaranteed by high availability and performance of the underlying software and hardware infrastructure. Proper planning and deployment of the computer systems and accurate assessment of the possible load and number of requests does not guarantee, solely, the long-term stability and high performance of the system, since the risks of the failures of same hardware and software constituent components is increasing in time. The disability of the hardware can be caused mostly by its ageing. However the failure of the software components might be related to the system's misconfiguration and unexpected load and also can be caused by the different security threats. All these factors finally have the negative influence on the quality of provided services. Thus stability of large computing system can be represented as a non-linear dependency of the critical components of the infrastructure to its serviceability. In order to make possible an application of the machine learning and statistical data analysis tools for modelling of such dependency, it is important to extract the quantitative measurements of particular metrics and status of the entire system. The best sources for such an information are the monitoring tools and the systems log files.
Tools and applications for the monitoring and checking the service availability can be classified in two major groups. Applications performing the service availability tests and methods analysing the service log information. Methods from both groups have their advantages and disadvantages.
Systems performing the tests for the service availability check are integrated in the infrastructure as monitoring tools. In case of the problems affecting the entire infrastructure, the capability of the monitoring tool to perform the tests and measurements are lost. Therefore usage of the quantitative monitoring information from such kind of sources needs the preprocessing step in order to filter out the outliers and the improper data. However advantage of such monitoring systems is provisioning of the highly granular quantitative information about the services and underlying systems, which makes possible to create the system model and identify its behavioural patterns.

The second group of tools and applications, analysing the system log information, nowadays are getting higher importance, since each system being a part of the complex computing and data storage infrastructure has its logging mechanism. By default the log information is available in a plain text format and contains number of the key words with the time stamps indicating the state of the service in a particular moment of time. Though for efficient usage of the log data, i.e. extraction of the useful information it is required to know the key words indicating the service failure or degradation. Nevertheless, there are risks of missing some critical information since the infrastructure malfunctioning also contains potential risks affecting the logging capability, i.e. there might be missing information about the failure or a even wrong time stamp for it.
Advantages and disadvantages of the service availability and disability detection approaches motivated to use the monitoring data of the services and underlying infrastructure together with the log information. The merged data gives sufficient information for performing further statistical analysis, with the goal of the system's failure modelling.
The basic workflow of the service status detection method, has been deployed and validated in the scope of this research. The process consists of several key steps including the service related and underlying infrastructure monitoring and log information aggregation, its preprocessing and statistical modelling (see fig. 7.1). The model derived based on ANFIS can detect the service availability and disability. Also, in the scope of the workflow it is possible to identify the root cause of the service related problems, which is possible by feature selection algorithm implemented using SVM method.

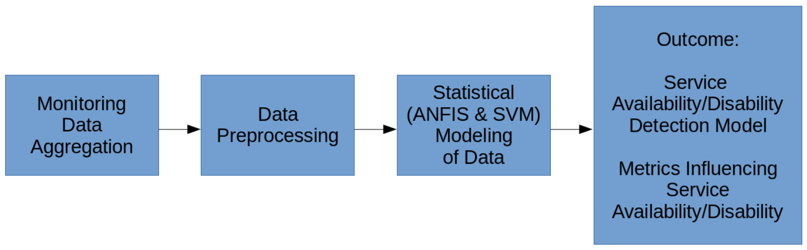

Monitoring Data Aggregation
Data Preprocessing
Statistical (ANFIS & SVM) Modeling of Data
Outcome:
Service Availability/Disability Detection Model
Metrics Influencing Service Availability/Disability

Figure 7.1.: Service status failure detection workflow.

7.1.1. Data Collection

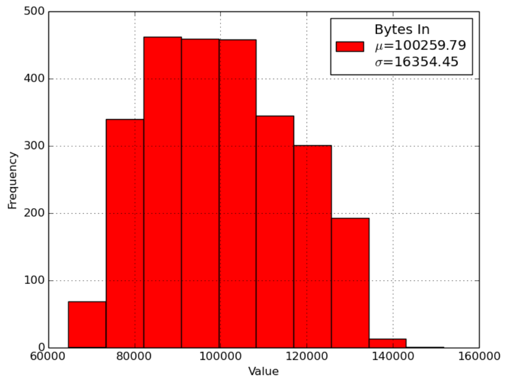
The case study was conducted in order to show the efficiency of the service status identification method based on ANFIS. The target of the case study was the Pnfs manager component of the GoeGrid dCache storage system. Choice of the Pnfs manager for the case study was motivated by its criticality for the entire storage system and due to the need of its failure reasons investigation. During the case study the underlying hardware and software infrastructure has not been changed, i.e. the data has been aggregated from the homogeneous environment, which allowed to perform the objective modelling of the Pnfs manager availability.
The main sources of the data for the case study were the Ganglia monitoring tool and

the dCache log file. The Ganglia monitoring tool consists of two major components, the server and the clients. Each server monitored by the Ganglia, hosts the Ganglia client service, which monitors and aggregates the cpu, memory, network, hard drive and the operating system related monitoring data and pushes to the server side. The pushed monitoring information is stored in the Round Robin Database (RRD) [243]. This information can be extracted in XML format, hence it is possible to use the data for further analysis. The dCache log data was also stored in a reusable format and in particular written in the plain text file. In both cases, for each entry in the RRD and log file, the corresponding time stamp has been defined, which made possible the synchronization and merging the data in the form of the following matrix (see 7.1.1.1). Each row of the matrix corresponds to the data synchronized according to a particular time stamp, where n corresponds to the total number of datasets (data vectors), k corresponds to the number of different monitoring metrics, $m_{1...n,1...k}$ represent the measurements of the monitoring metrics from Ganglia and s_n stands for the status of the service. In this case the service status corresponds to the Pnfs manager availability or disability and it is a binary value, where 1 indicates that the Pnfs manager was functional and 0 indicates the opposite.

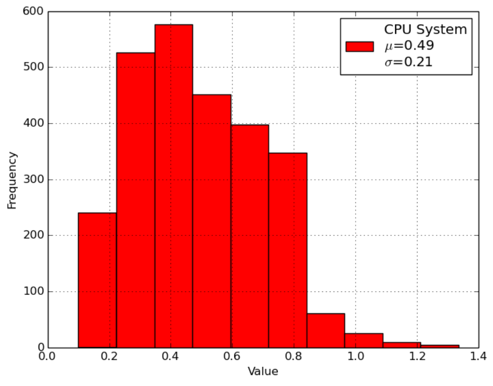
$$\begin{pmatrix} m_{11} & m_{12} & ... & m_{1k} & s_1 \\ m_{21} & m_{22} & ... & m_{2k} & s_2 \\ m_{31} & m_{32} & ... & m_{3k} & s_3 \\ . & . & ... & . & . \\ . & . & ... & . & . \\ . & . & ... & . & . \\ m_{n1} & m_{n2} & ... & m_{nk} & s_n \end{pmatrix} \tag{7.1.1.1}$$

Data extraction and processing from the RRD as well as from the dCache log file has been performed using the scripts developed on the python [162] programming language in scope of this research project. The key aspect in terms of the dCache log information processing was its filtering for the period of the case study. The filtering has performed according to the following key phrases *"namespaceDomain","No Route to cell for packet"*, which were identified based on the experience and the dCache system administrator's documentation. Presence of the combination of these key words in a single line of the dCache log data file indicated the Pnfs manager service failure and their absence its availability. Presence of the time stamp for each entry in the dCache log file allowed to identify exact moments of the service failures as well as the periods of its availability.
During the period of the data collection, filtering and synchronization 2639 datasets were aggregated, where only 16 of them were corresponding to the service failures. Thus, the aggregated data was extremely unbalanced in favour of the service availability.
Due to the different scales and units used for the number of monitoring metrics the data normalization was required (see fig. 7.2). To keep the statistical representativeness of

the data, the method of statistical normalization [244] has been applied. The statistical normalization, which is also known as z transformation method, transforms the data such that its mean is 0 and the variance equals to 1, more precisely the normalized, Z, values are derived using the following formula $Z = \frac{X-\mu}{\sigma}$, where X is the initial dataset, μ and σ are mean and variance correspondingly. Each column of the aggregated data matrix 7.1.1.1 has been normalized using the z transformation method, implemented in the RapidMiner tool [245] (see fig. 7.3). The normalized data has the negative values and 99% of entire data is accumulated in the interval of $[-3, 3]$. Since the monitoring metrics does not have the negative values one more step of the data transformation has been applied. In particular the data values has been increased by constant, equal to 3 (see fig. 7.4). Described data transformations correspond to the data pre-processing step from the approach perspective, which is unavoidable before the start of statistical analysis. The key aspect at this step is avoiding the loss of the information provided by the data and keeping the values in a positive range, since all monitoring metrics are non-negative real numbers by their definition.



(a) Bytes In metrics.

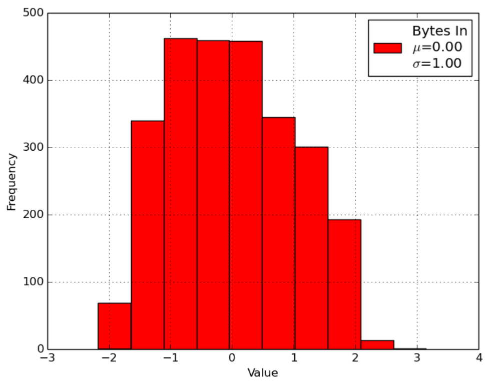


(b) CPU System metrics.

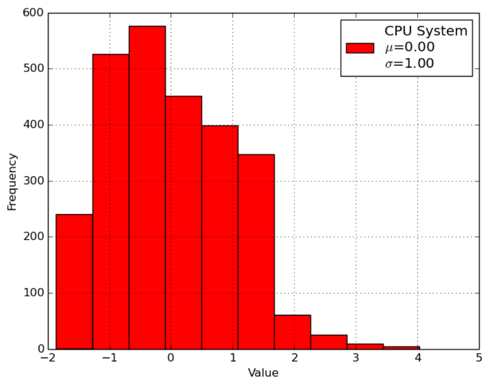
Figure 7.2.: Statistical representation of initial monitoring metrics from Ganglia. Difference between scales ranges to 10^6.

7.1.2. Application of the Linear SVM Algorithm for the Service Status Identification

Application of the machine learning and statistical data analysis algorithms for the large scale computing facilities monitoring data processing requires large computational and memory resources. The high demand on the computational resource is caused by the large number and size of the monitoring metrics and the corresponding information. Therefore, it is important to identify the optimal algorithm and methodology in order to make the approach applicable in practice.



(a) Bytes In metrics.



(b) CPU System metrics.

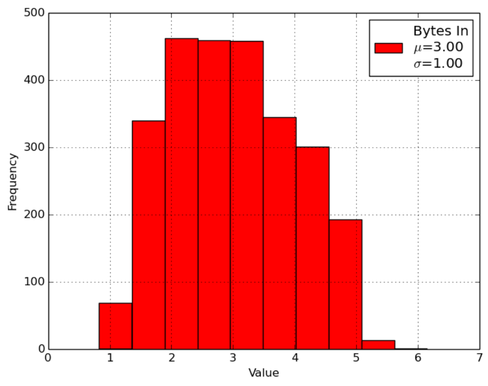
Figure 7.3.: Statistical representation of z transformed monitoring metrics from Ganglia.

After the data pre-processing and its representation in the form of the matrix 7.1.1.1 the status identification process can be considered as a binary classification problem, where the classes correspond to the available and malfunctioning statuses of the service. Thus, it is required to identify the methodology for analysing the system monitoring and log information against failures, which might lead to the model for the quick and precise service failure identification.

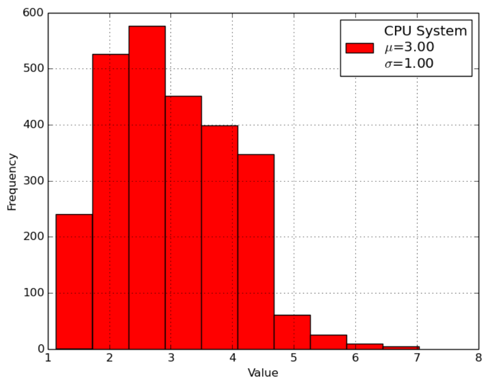
The binary and multi-label classification problems [246] are widely explored and there are available number of approaches for their solution. Among the methods, SVM is one of the actively used and highly efficient approaches, which is considered as an alternative to ANN based methods developed for the classification problems. Thus, in the scope of this case study ANFIS based approach will be compared to the SVM efficiency.

During the case study, for applying the SVM classification algorithm the RapidMiner tool was employed [247]. The key parameters of the SVM are the *kernel type* and the *Complexity Constant - C*. In this case study these parameters are set to the linear kernel and $C = 0$. Application of linear kernel requires the smallest computational power and memory among other available kernels, hence it can be implemented in practice efficiently. The C parameter defines the level of SVM model generalization and since the ultimate goal is to have the generic model, the value of C parameter has set to the possible minimum, i.e. 0.

Another goal of the case study is the service failure root cause identification, which can be interpreted as a search of the particular monitoring attributes influence rate on the service status. Thus it is required to identify the contribution of each component to the service status. Among the number of techniques for the solution of such type of problems, the feature selection algorithm based on SVM [248] is identified as one of the most efficient. Implementation of the SVM based feature selection in RapidMiner (see Appendix A) is completely independent from the SVM based classification method. Application



(a) Bytes In metrics.



(b) CPU System metrics.

Figure 7.4.: Statistical representation of the monitoring metrics from Ganglia after z transformation and shifting to the positive range.

of the feature selection method gave the following results (see fig. 7.5). Thus based on the feature selection results and the known semantics of the monitoring attributes it is possible to identify the the most influencing components for the service status, i.e. it is possible to perform the service disability root cause analysis. According to the result of the feature selection, three components the "Load Fifteen"[1], "CPU System"[2] and the "TCP Established"[3] has the highest importance for the service status, however the memory consumption is not critical at all. Based on the provided information and the system administration experience it was obvious that the high load on the system was caused by the large number of requests coming via the network, which finally was finally causing the system's disability. Based on this conclusion and investigation of the Pnfs manager's tuning parameters it has been revealed that the bottleneck was the meta data database. Due to the poor performance of the Pnfs manager meta data database the requests arriving from entire storage system for accessing the data were not served with the sufficient speed, which was causing the performance problems for the entire data storage infrastructure.
The feature selection results also identified the most valuable monitoring attributes influencing the service status. Thus, the possibility of the input data dimension reduction, i.e. taking only three most important attributes has been identified. Reduction of the input data dimension, i.e. in this case using $k = 3$ for the 7.1.1.1 matrix allowed to increase the methods applicability in practice.
Results of applying the SVM[4] are presented in terms of the contingency table (see table 7.1). The table shows overall SVM accuracy and its capability of adequate detection

[1]Average load in 15 minutes
[2]CPU time consumption by the processes
[3]Number of established TCP connections with the host
[4]For RapidMinder stream see Appendix A

of the service status when it was available and malfunctioning. Two columns with the title "True 1" and "True 0" represents the actual status of the Pnfs manager, while rows with the title "Prediction 1" and "Prediction 0" corresponds to the classifier predictions regarding the service status. The class recall for "True 1" and "True 0" represents the classifier accuracy for correct detection of each of the status, i.e. the percentage of the correctly predicted statuses of the service. Since the objective here is to identify the system failure, i.e. the rare events the value of class recall for "True 0" shows the disability of SVM for identification of the malfunctioning status of the service. In order to validate the sufficiency of the application of three most influencing attributes as input, a number of different experiments with different sets of input data were conducted (see Appendix B). The SVM classifier has been applied to all 18 attributes, also for the extraction of more information there were grouped all possible combinations of two most important attributes out of three together with their gradients. The gradient in the particular case study was the difference of two subsequent monitoring metrics measurements [5]. However the recall for "True 0", for all input data combinations was 0%.

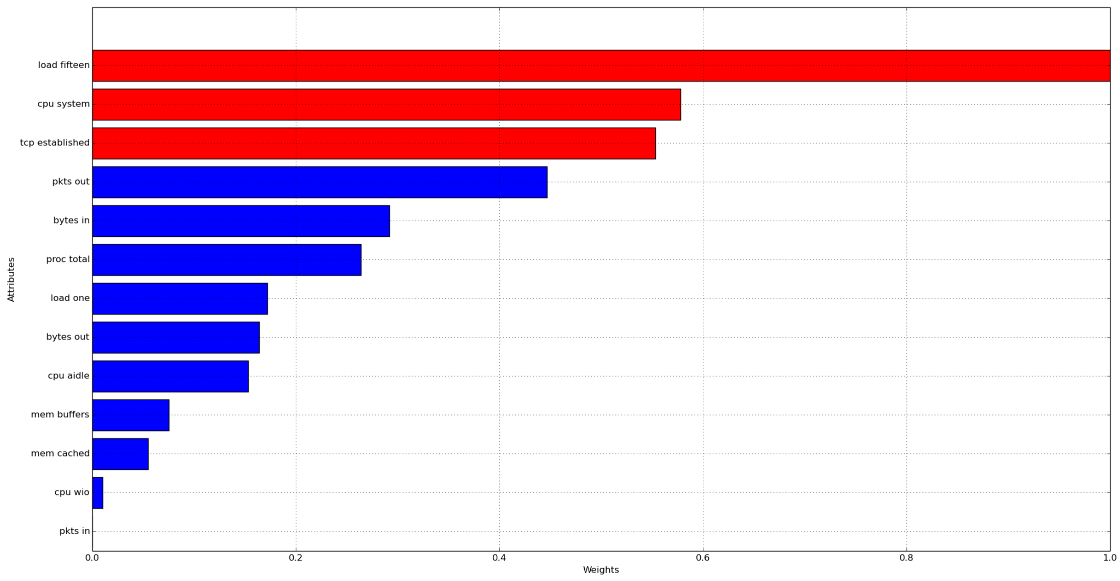


Figure 7.5.: Attribute ranking according to their influence on the process state.

7.1.3. Application of ANFIS for the Service Status Identification

ANFIS represents the classification techniques, which is considered to be an alternative for the Support Vector Machines approach. The core feature of ANFIS is its learning capability, which is implemented using the Feed Forward Neural Networks training algorithm, described in chapter 6.

[5] $m_i j - m_{i-1} j$

	True 1	True 0	Class Precision
Prediction 1	2623	16	99.39%
Prediction 0	0	0	0.00%
Class Recall	100.00%	0.00%	Accuracy: 99.39%

Table 7.1.: SVM Efficiency with the three most important attributes as input.

Evaluation of ANFIS approach required splitting the entire data in two parts, the training and the testing. Based on the training dataset and the ANFIS learning procedure, all the parameters of activation functions as well as the parameters for the membership functions has been identified. Thus at the end of the ANFIS training process the service failure identification model has been derived, which was evaluated on the testing data set. Evaluation procedure meant the calculation of the contingency matrix[6], which was performed by the comparison of the actual service statuses and ANFIS outputs provided from the testing data.

Input data for ANFIS, as well as in case of SVM consisted of three most important attributes identified by the SVM based feature selection procedure and it has been split into two equal parts for training and testing. The data split has been performed using the stratified sampling algorithm [249]. The stratified sampling method guarantees the similar class distributions in training and testing data sets. The stratified sampling approach has been applied also for during the SVM evaluation (see Appendix A). Thus results derived from ANFIS testing phase can be considered as general performance of ANFIS.

The key parameters for ANFIS, that needed to be known in advance, were the number and the type of the membership functions for each input variable. Here, three membership functions were used per input variable and in this way each of the membership function has been associated with three possible states of the monitoring metrics, "OK", "WARNING" and "CRITICAL". These states were widely used in a very basic monitoring tools and clearly represents the semantics of the particular metrics value. As for the type of the membership function, the decision was made in favour of Gaussian membership function. The decision has been motivated due to the absence of the linear relationship of the input data and the service status. The output membership function had the constant parameters, i.e. the first order Sugeno model was applied. Overall training process consisted of 100 iterations, i.e. 100 epochs. The result of the ANFIS training process in terms of the training error is shown on the following plot (see fig. 7.6). The accuracy, precision and the recall of the trained ANFIS model is shown in the contingency matrix (see table 7.2).

According to the ANFIS contingency matrix the overall accuracy of ANFIS is 99% but the significant improvement can be observed in terms of the correct identification of the

[6]e.g. see table 7.1

failed service status. Correctly predicted five actual failures from eight[7] in total allows to consider the ANFIS as sufficiently reliable for detection of a rare events such as service failures. Thus, ANFIS recall for the failure prediction is 62.5% and the class precision is 71.43%, these results mean that there were only very small number of false alarms[8], which gives the significant reason to rely on the method's output. The ANFIS based approach has been tested for various combinations of the input variables (see Appendix C) and the best results can be observed when the three top-most attributes are used together as an input. ANFIS training process for all 18 attributes takes large amount of computational power and time, therefore it was not even considered for the analysis.

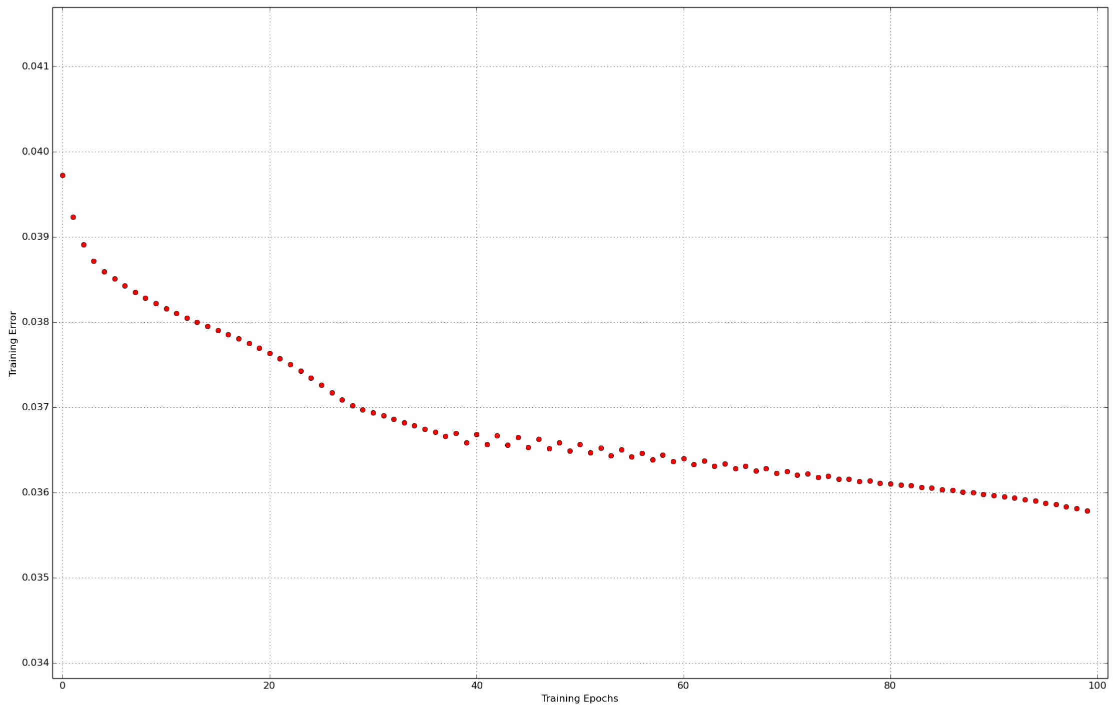


Figure 7.6.: ANFIS training error for 100 epochs.

All the derived results were generated using the Fuzzy Toolbox package [250] of the MatLab software [251]. Parameter set of trained ANFIS is available in appendix D, while the ANFIS structure and the set of rules are presented here (see fig. 7.7 and 7.8).

Thus, even in case of extremely unbalanced data, the ANFIS based model is able to detect the rare occurrences with significant efficiency and small amount of consumed computational resources. This facts allow to consider it, as a good candidate for the

[7]The total number of service failures is 16, half of it was used for the training and another half for testing.

[8]Only two according to the table 7.2

	True 1	True 0	Class Precision
Prediction 1	1310	3	99.77%
Prediction 0	2	5	71.43%
Class Recall	99.84%	62.50%	Accuracy: 99.62%

Table 7.2.: ANFIS Efficiency with the three most important attributes as input.

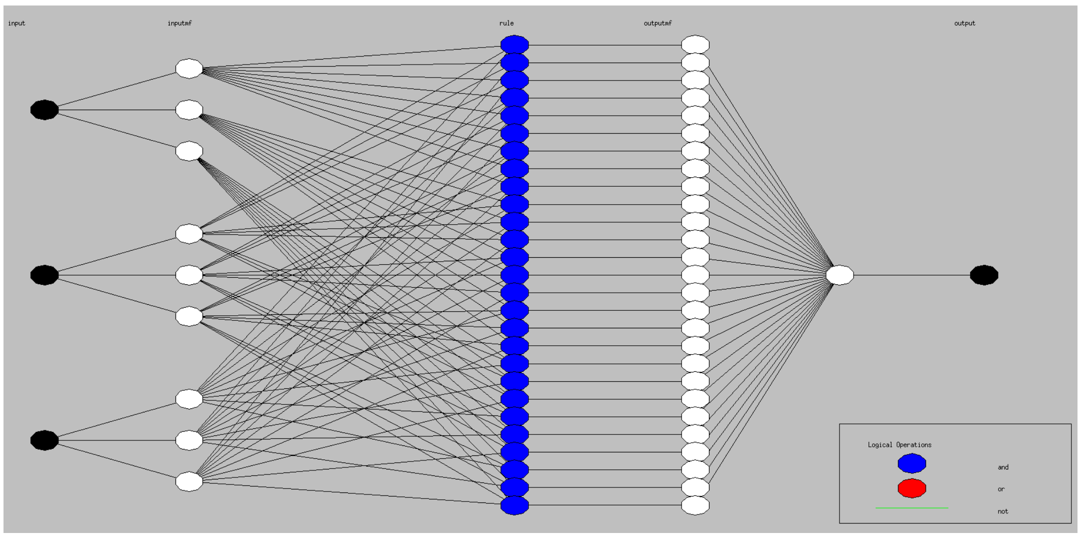


Figure 7.7.: ANFIS structure.

implementation in modern monitoring applications as one of the efficient data analysis and machine learning approach.

7.2. The Service Response Time Prediction

Service status identification using the machine learning and statistical data analysis techniques is an efficient and reliable way for the service failure detection in a large scale computing infrastructures. However, it requires aggregation of significant amount of the balanced monitoring information, i.e. data with the significant amount of records describing the systems parameters in the phases of its availability and disability. Without such an information it is not trivial to get the highly accurate system failure detection model. Although, quick, reliable and accurate service status and its failure root cause identification is important for the systems administrators, the prediction of the service related risks is highly efficient for the implementation of the automated problem mitigation approaches.
This problem of the in advance detection of the service failure or degradation can be considered as the failure or disaster prediction problem for non-linear systems. The key

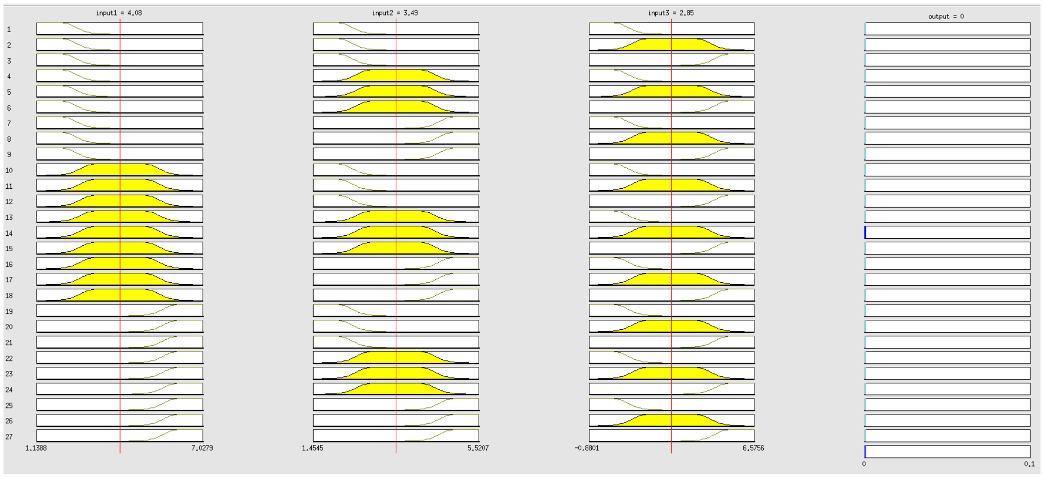


Figure 7.8.: ANFIS set of rules, 27 in total.

characteristic for the service availability degradation and also for the detection of its disability is the Service Response Time. The SRT also includes the information about important hardware or software parameters, hence it can be used both as a symptom and an indicator of the service failure. Thus SRT can be used as a source for the analysis process and as a prediction target, which allows to detect the service related problems in advance.

7.2.1. Time Series Analysis and Input Data Structure Identification for ANFIS

One of the widely adopted techniques for the forecasting of continues values, is the time series analysis[9]. Time series analysis for a particular process relies on the knowledge of the data from the past[10], i.e. on the data, which contains the information about the different patterns of the process dynamics. Methods applied to the time series prediction problems are focused on the generalization of such patterns in the scope of a single model, i.e. single set of equations describing the service behaviour according to the values of the key characterizing parameters. Historical information here, is a sequential collection of data, D_i, with its starting point at t_1 and its final value at the moment t_i, where $t_{j+1} - t_j = p, \forall 1 \leq j \leq i-1$, the p here represents the equal time intervals between the SRT measurements. The $D_i = \{d_j\}, 1 \leq j \leq i-1$, where d_j is the value of the SRT at the moment t_j and the elements of D_i are ordered in sequence such that $t_j < t_k, \forall j < k$.
Under the assumption that D_i is known, the aim of the time series prediction model, M, is to provide a forecast for d_{i+n}, where $n \times p$ represents the prediction time interval

[9]Or time series prediction.
[10]Frequently called historical data.

in minutes. In other words the time series prediction problem can be formulated in the following way: $d_{i+n} = M(D_i) \pm e_{i,n}$, where $e_{i,n}$ represents the prediction error, which is increasing with n, indicating the length for the prediction time interval.
Time series analysis, or prediction, is also related to the pattern detection in data, which is impossible without taking into account its qualitative and quantitative aspects. ANFIS based approach, due to its characteristics, has been efficiently used for handling such type of challenge. The structure of ANFIS for the time series prediction is shown on the following scheme (see fig. 7.9). The ANFIS input has been organized as follows, the $g_3(SRT_i)$ was the SRT measurement at the time moment i, $g_2(SRT_{i-1}, SRT_i)$ corresponded to the $\frac{SRT_i - SRT_{i-1}}{2}$, where the SRT_{i-1} was the response time measurement one time interval before the time moment i. The $g_2(SRT_{i-1}, SRT_i)$ input component also has been considered as an analogy to the first order derivative, since the sign of the difference indicated the increase or decline tendency of the SRT metric. The third component of input $g_1 \frac{SRT_i - 2SRT_{i-1} + SRT_{i-2}}{4}$ was an analogy to the second order derivative. The SRT_{i-2} corresponded to the response time measurement at the time moment $i-2$, i.e. two time intervals before t_i. Merging the g_1, g_2 and g_3 input data components led to the following input vector $l = (\frac{SRT_i - 2SRT_{i-1} + SRT_{i-2}}{4}, \frac{SRT_i - SRT_{i-1}}{2}, SRT_i)$, and it has been supplied to ANFIS as an input. The output of ANFIS was the SRT_{i+1}, i.e. the response time prediction in one time step forward. Efficiency of the input data arrangement in the form of the l vector is shown on the following plots (see fig. 7.10). For generalization of the input data structure efficiency check results, the core parameters of the input variables, i.e. the membership functions, were varying from three to five. Experiments has also shown that assignment of more than five membership functions per input variable caused significant growth of the computational resource consumption and time, hence made impossible the usage of approach in practice. According to the results 7.10 the MSE and RMSE measurements are less in case of ANFIS trained with the input data, structured identically to the l vector. This results, helped in definition of the input data structure for the case studies, described in the further subsections.
As an input for the experiments the SRT data has been aggregated from the GoeGrid infrastructure and in particular from its storage and batch systems. The data aggregation process was ongoing inside the infrastructure, which excluded the influence of external factors on the data collection process. In addition, there were defined quality indicators for excluding the data containing the subjective information. The data aggregation process is depicted on the following scheme (see fig. 7.11) and the approach was based on sending the plain service requests and recording the time required for the response. The target services for the case studies were the dCap service functioning on a particular hardware server and the "pbs server" service, guaranteeing the batch system functionality.

7.2.2. Application of ANFIS for the dCap Service Response Time Prediction

During the case study, the dCap service has been used most frequently for accessing the experimental data, stored on the GoeGrid dCache storage system. It is important

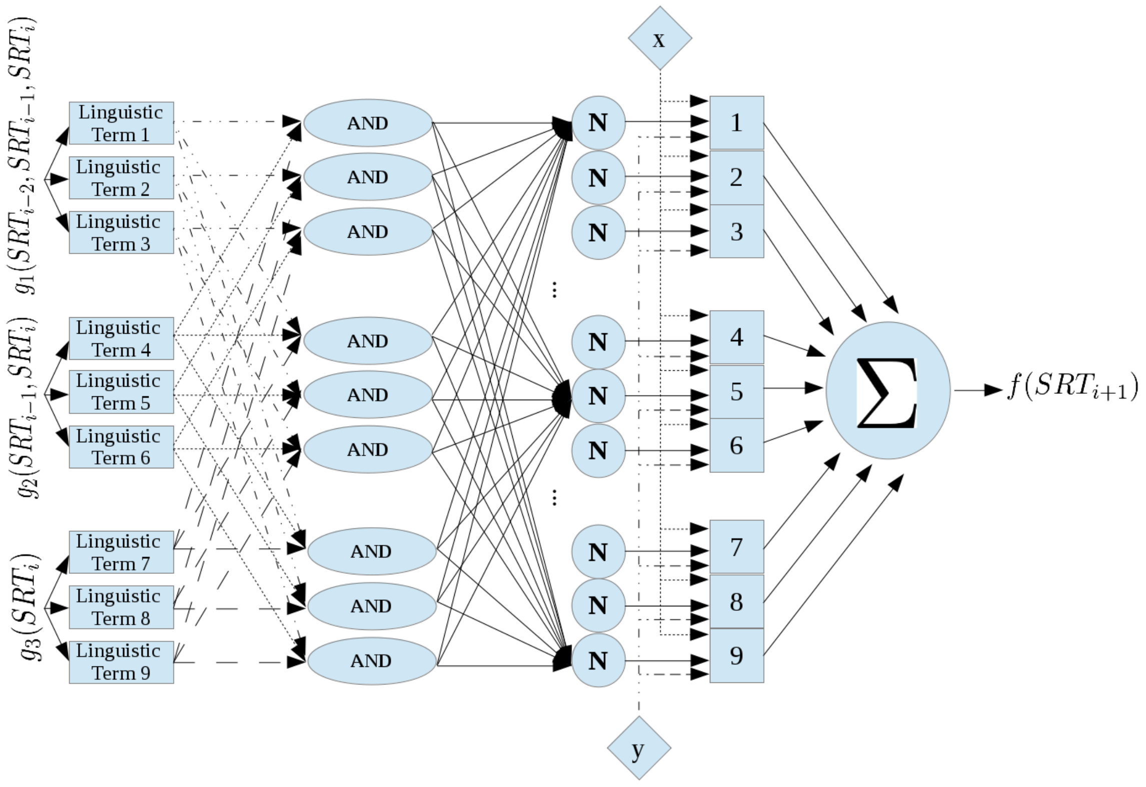
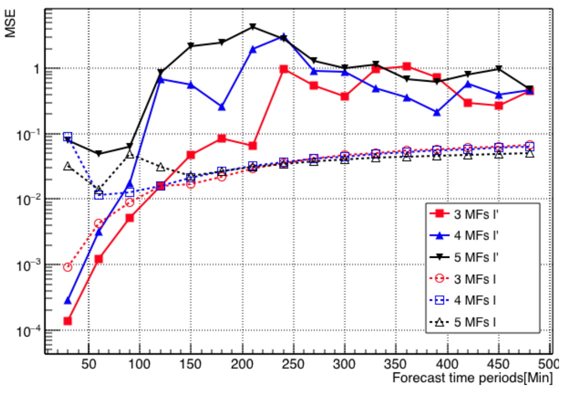
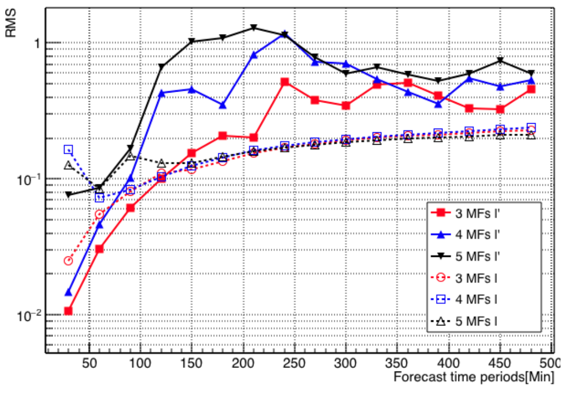


Figure 7.9.: ANFIS structure for the time series prediction.

to admit that GoeGrid provided dCap services via several dedicated servers, however the implemented load balancing mechanism guaranteed the equal distribution of load on all of the dCap service provider hosts. Thus the SRT data from different dCap service providers can be considered as homogeneous, i.e. the results can be generalized for all dCap service providers at GoeGrid.
The SRT data aggregation process was running in every six minutes according the best practices from the Ganglia monitoring tool. Thus one step ahead prediction indicates the SRT forecast in six minutes. Predictions for the longer period of time is based on application of already predicted values as an input. Thus, prediction of every next SRT value is less accurate and it is important to define the adequate time frame for the forecast. In this case study the prediction varies from six minutes to eight hours, since knowing the service state in eight hours can be considered sufficient time for proper action planning.
For identification of the general set of parameters for ANFIS, which could led to the best performance, the number of membership functions for each variable varied from three to five. For the accuracy measurement the Root Mean Squared Error (RMS), Mean Squared Error (MSE), Absolute Error and the Mean Absolute Percentage Error (MAPE) has been used. For more precise visualisation of the results the Regression Er-



(a) Mean Squared Error.



(b) Root Mean Squared Error.

Figure 7.10.: ANFIS model accuracy in case of two different arrangements of the input vector.

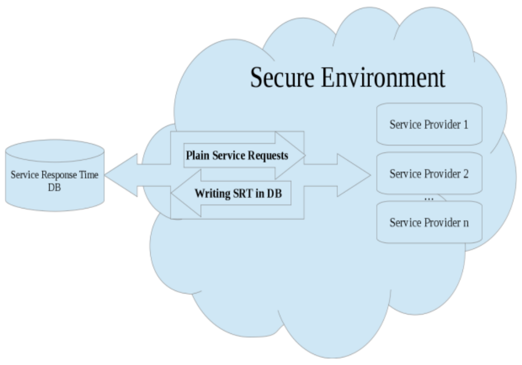
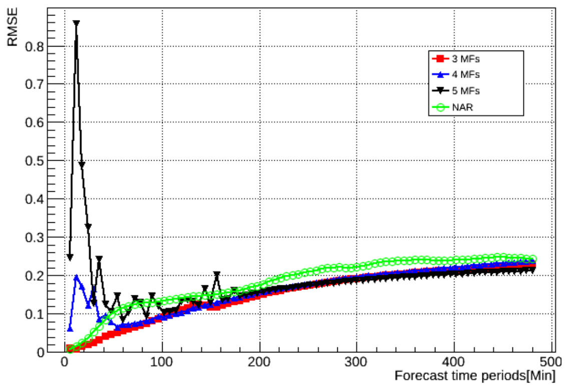


Figure 7.11.: Service Response Time data aggregation process in the GoeGrid (secure) environment.

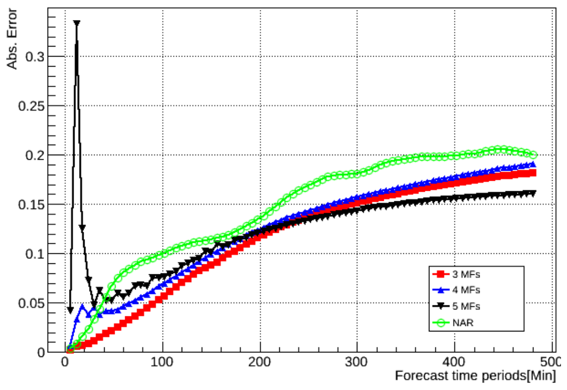
ror Characteristics (REC) curves [252] were employed for the MAPE metrics. Here, the REC curves show the regression method performance using the MAPE error metrics. As it is shown here, the REC plots contain the ANFIS based method accuracy assessment by calculating the difference of the predicted and actual SRT values for the predictions from six minutes up to eight hours[11]. According to the prediction accuracy plots (see fig. 7.12 and 7.13), performance of ANFIS based model with three membership functions per input variable is the best up to three hours and changes in favour of the model with five membership functions for the SRT predictions further, till eight hours. For the complete picture, the ANFIS accuracy is also compared with the Nonlinear Autoregressive Neural Network (NARNET)[12] [253]. The NARNET is a widely used technique for the time series prediction. However the results has shown its comparable to ANFIS, but less

[11]8 hours correspond to 480 minuts.
[12]Green marker on the plots

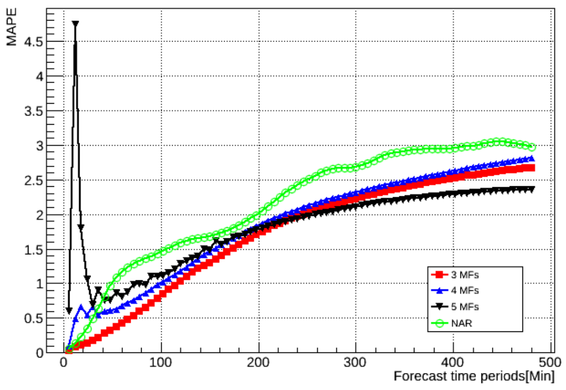
accurate performance. Especially it is well depicted on the following plot (see fig.7.13).



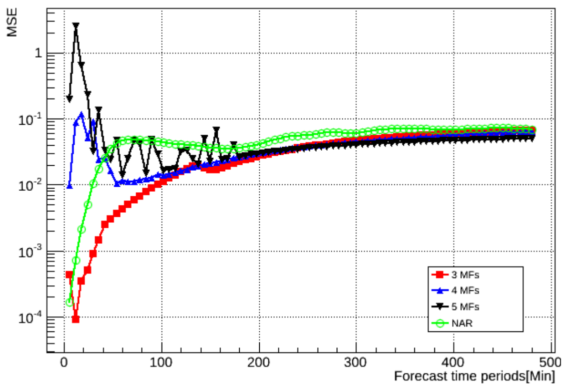
(a) Root Mean Squared Error.



(b) Absolute Error.



(c) Mean Absolute Percentage Error.



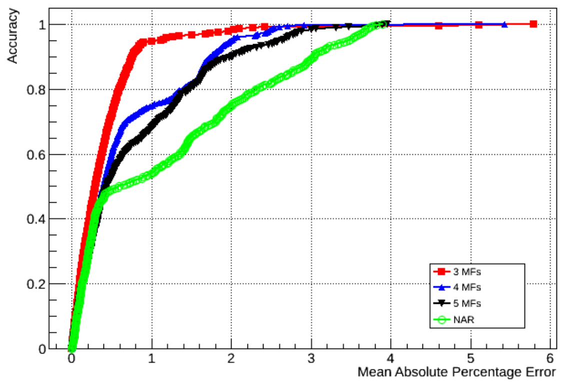
(d) Mean Squared Error.

Figure 7.12.: ANFIS based prediction accuracy assessment metrics.

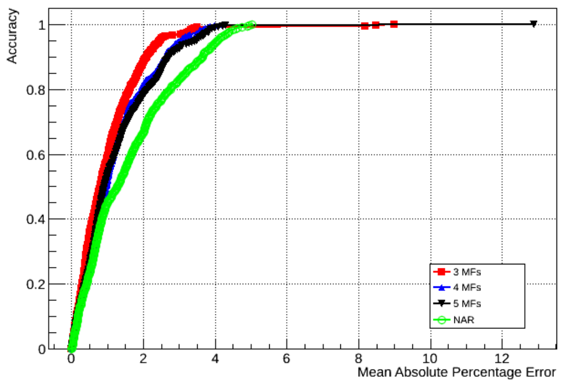
Based on the ANFIS assessment results, it is critical to identify the maximum error margin for the longest prediction, which in this case is the forecast for eight hours. The results of ten fold cross validation study and comparison of ANFIS with NARNET are depicted in the table (see table. 7.3), which shows general advantage of ANFIS with five membership functions for making the SRT assessments up to eight hours.

7.2.3. Application of ANFIS for the Batch System Service Response Time Prediction

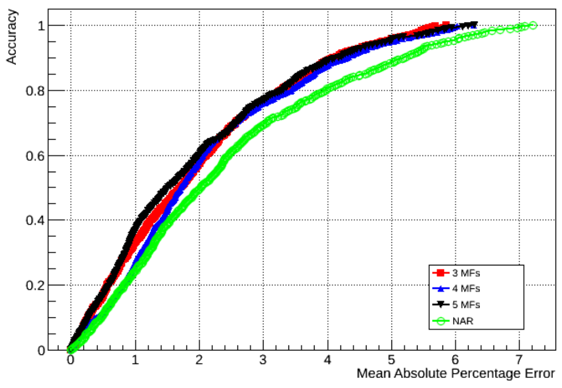
Proper functionality of the Torque batch system, used at the GoeGrid facility, requires the optimal performance of the ”pbs server” service. This service is the core component of the batch system, which communicates with all compute nodes and the scheduler,



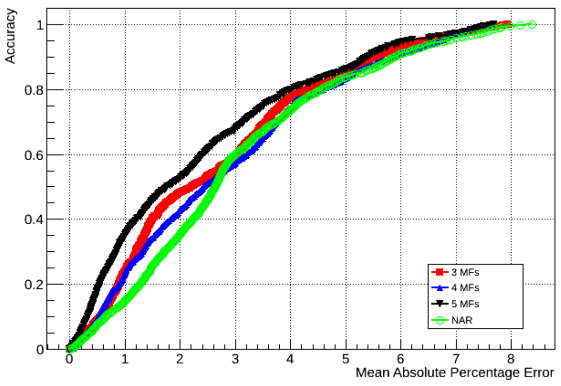
(a) MAPE for one hour.



(b) MAPE for two hours.



(c) MAPE for four hours.



(d) MAPE for eight hourse.

Figure 7.13.: Regression Error Characteristics (REC) curves for MAPE.

therefore high load on the system, which can be caused by the submission and management of large number of computational jobs can affect the SRT of the ”pbs server”. The high speed communication between the ”pbs server” and the batch system scheduler is critical for the optimal consumption of the available computational resources, therefore these two components of the batch system are hosted on the same server, i.e. the CPU and the memory consumption is shared among the ”pbs server” and the scheduler services. Thus SRT prediction of ”pbs server” can be used for assessment of the performance of batch system's central components, which is critical information for the risks proactive management among the entire infrastructure.
Similarly to the dCache system, the torque batch system provides the CLI for the communication. This fact allowed to collect the SRT data using the same approach as for the dCap service, i.e. sending the plain requests and recording the time required for the response. For this case study as well as in case of the dCap SRT prediction, the structure of the input data for ANFIS was identical to l vector.
Results for the batch system service response time prediction are also based on the ten

Methods	100% - MAPE
ANFIS 3MF	97.3%
ANFIS 4MF	97.2%
ANFIS 5MF	**97.7%**
NARNET	97.0%

Table 7.3.: Comparison of ANFIS and NARNET MAPE metrics for eight hours forecast for dCap service. MF, Membership Function.

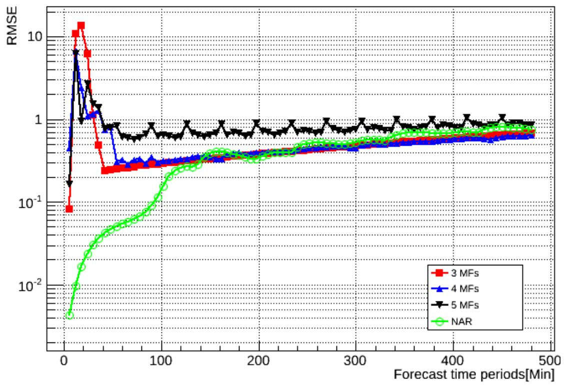
fold cross validation and the prediction interval varies from six minutes to eight hours. As in case of the dCap service case study, the NARNET is considered as an alternative method to ANFIS.

Here, the number of membership functions for each of the input variable also varies from three to five. According to the ten fold cross validation results (see fig. 7.14) the best performance, for short and long term predictions can be achieved when four membership functions are employed for each of the input variable. Here as well as in case of the dCap SRT prediction case study and for the Pnfs manager status identification the Gaussian membership functions were employed.
Distinguished accuracy of ANFIS can be also observed from the REC plots of the MAPE error metrics (see fig. 7.15). Though, for the short term predictions, up to one hour, MAPE of NARNET was less than in case of ANFIS, but for the long term predictions the performance of ANFIS was significantly better, which is also shown in the following table (see table. 7.4).

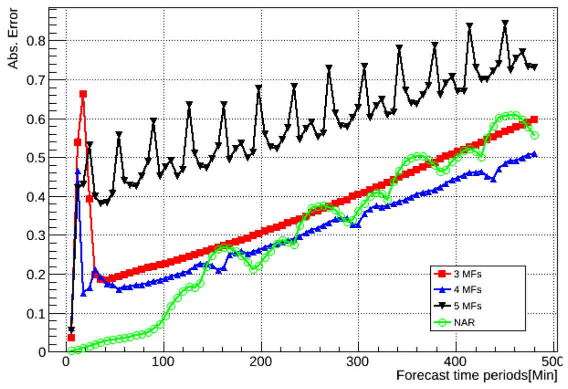
Methods	100% - MAPE
ANFIS 3MF	90.4%
ANFIS 4MF	**93.6%**
ANFIS 5MF	84.6%
NARNET	88.2%

Table 7.4.: Comparison of ANFIS and NARNET MAPE metrics for eight hours forecast for "pbs server" service. MF, Membership Function.

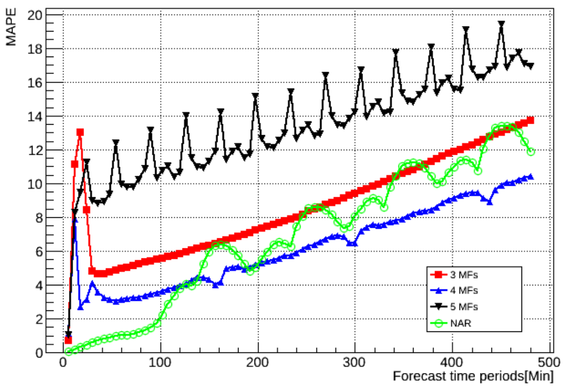
Case studies and the approaches described in this chapter were presented on international symposium and conference. The service status identification and the failure root cause analysis approach has been presented on the "XXIV International Symposium on Nuclear Electronics and Computing" in Bulgaria, 2013 [254]. The service status prediction approach and the results has been shown as a poster presentation on "21st International Conference on Computing in High Energy and Nuclear Physics" in Oki-



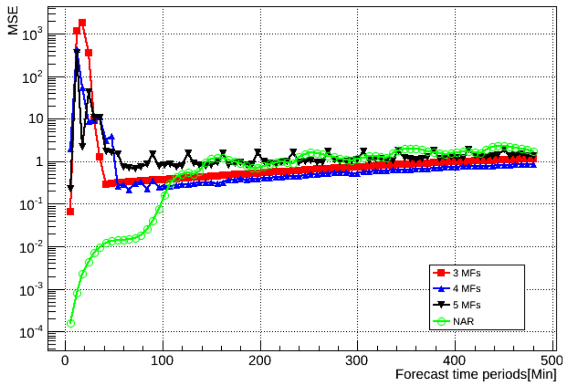
(a) Root Mean Squared Error.



(b) Absolute Error.



(c) Mean Absolute Percentage Error.



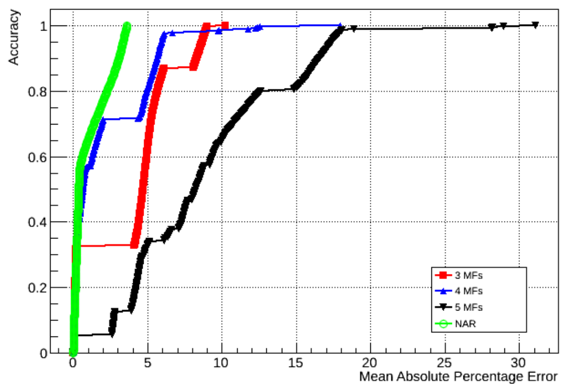
(d) Mean Squared Error.

Figure 7.14.: ANFIS based prediction accuracy assessment metrics.

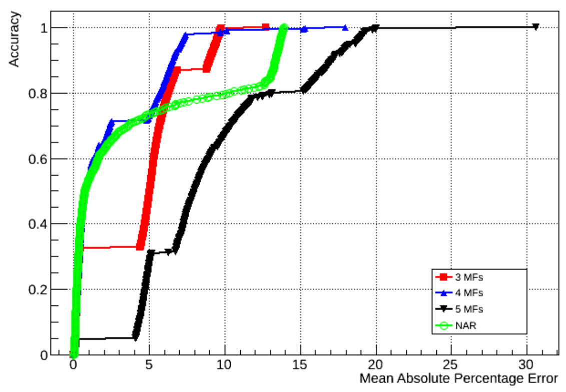
nawa, Japan, 2015[13] [255]. The publications corresponding to the both presentations has been entered in the peer reviewed conference proceedings. For the work describing the service status identification and the failure root cause analysis approach the title of the paper was "The Smart Monitoring System for the Grid Site" [256] and in case of the service status prediction the title of the proceeding was "Automation of Large-scale Computer Cluster Monitoring Information Analysis"[14].

[13] April 13-17

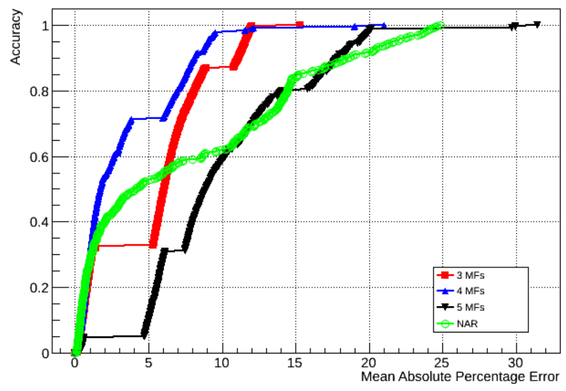
[14] To be published in the Journal of Physics: Conference Series (JPCS), which is the part of IOP Conference Series



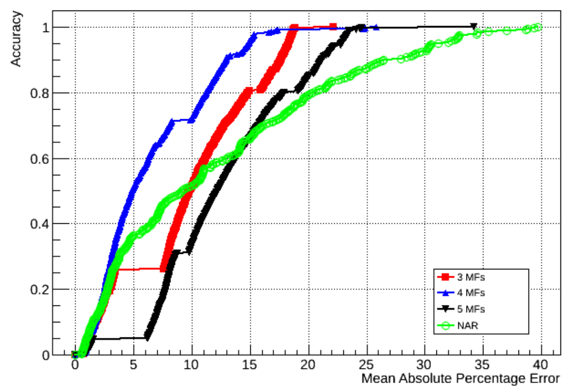
(a) MAPE for one hour.



(b) MAPE for two hours.



(c) MAPE for four hours.



(d) MAPE for eight hourse.

Figure 7.15.: Regression Error Characteristics (REC) curves for MAPE.

8 Summary, Discussion and Outlook

This chapter summarize the thesis and provides the discussion and outlook of achieved results and their potential applications.
The first subsection is dedicated to the short summarizing overview of the previous chapters and achieved results. The next subsection is dedicated to the discussion of the results and the approach limitations. The last subsection describes the scheme of integration of described approaches in the large scale computing facility and its possibilities of its development.

8.1. Summary

Motivation of this research was efficient application of the monitoring data for automatic detection of the problems at the GoeGrid facility. The purpose of the chapters 2, 3 and 4 was an explanation of need, complexity and the structure of the GoeGrid resource centre and the WLCG infrastructure. Without WLCG computing platform and in particular, without the ATLAS distributed computing, provisioning of the computational power and the storage capacity to the ATLAS experiment would be impossible.
During the period of administration of the GoeGrid computing resource centre, the performance capacity and the importance of each subsystem has been studied. Chapter 4 is dedicated to the detailed description and summary of the GoeGrid components, capacity and their functionality.
GoeGrid, similarly to other resource centres consists of components prone to failure. Hence, number of different types of failures and problems has been observed during different periods the study. Some of the issues were caused by obvious problems, like hardware components failures or by the sudden shrinking of the free space on a particular system partition. However, most of the problems required significant fraction of time and resources, first of all for understanding of the root cause and after for its resolving.

During the case studies part of the management and administration processes of GoeGrid was governed by the automatic infrastructure management tool CFengine[1]. This tool was under the constant adaptation for automatic detection and solution of obvious problems. This was mostly possible due to the straightforward description of the problem cause, its detection rule and the required actions. However, translation of most of the complex problems detection and solution procedures in terms of CFengine rules was not possible.
After observation and analysis of the process for the computing infrastructure problem detection, it was clear that the first sign of the service or entire system disability was the sequence of the failed monitoring tests, which were constantly running by the internal and external monitoring systems. Details about the testing and monitoring infrastructure of GoeGrid is covered in chapters 3 and 4. Though, the problem detection process was clear, in most of the cases it was not enough for the quick identification of the problem source, hence for its elimination. The most complex work, for the problem resolution, was related to the investigation of the corresponding monitoring and log information. Thus, the key aspect for the problem source detection process automation was simulation of the monitoring and log data analysis, done by the system administrators. This observation and the review of the corresponding literature brought to the concept of Autonomic Computing, which was proposed by the IBM. The concept is explained and covered in details in the chapter 5. The core part of Autonomic Computing, that has been actively used in the scope of this research, was the self-awareness, i.e. the idea of automated detection and solution of the complex problems appearing inside the system. According to the Autonomic Computing architecture the key aspect for building the self-awareness functionality is the statistical analysis of the available monitoring data. Based on the review of the literature, which is provided in the first sub section of the chapter 6, one of the basic methodologies for the monitoring data analysis are the statistical and machine learning techniques used for the classification problems. This fact helped in identification of the next step, which was related to the search of an adequate techniques for performing the human like reasoning with the capabilities of extraction of both, the quantitative and qualitative information from the available data. Another key requirement for such an approach was optimal consumption of the computational resources during the data analysis. All these requirements were satisfied in case of ANFIS, which is the fundamental component not only for the service status identification but also for the prediction of its state for the period of eight hours. Important role in optimization of the ANFIS performance had the feature selection procedure, which helped to reduce the number of input data provided to ANFIS. The feature selection technique is based on the Support Vector Machine algorithm identifying the weight of each attribute influence on the status of the corresponding service. The SVM based algorithm used for the feature selection is different from the method used for the classification problem. For clearance of the used approaches the mathematical foundation of ANFIS and SVM functionality and their properties are discussed in chapter 6.
Chapter 7 is dedicated to the case studies, which were conducted for checking and vali-

[1]See chapter 4.

dation of the ANFIS based approach agains SVM and NARNET. The summary of the results is provided in the following subsection.

8.2. Discussion

Results derived using the SVM has shown 99.39% of accuracy for the Pnfs manager status identification case study. However the goal of the approach application is the identification of the rare events such as service disability. Thus, the more important metrics for the SVM based method assessment are the class recall and the class precision for the failures[2]. In case of SVM these metrics does not show the sensitivity of the model to the failure detection, at all. However, model demonstrated high performance for the detection of the functional state of the service. As an input data for SVM the three most important monitoring metrics were selected. The attribute importance calculation has been performed using the feature selection algorithm based on SVM. According to the results of the feature selection, the largest weights[3] were assigned to the three monitoring attributes, "Load Fifteen", "CPU System" and "TCP Established". These three metrics revealed the root cause of the system disability and its poor performance. The high load was mostly caused due to the large number of the requests coming on the Pnfs manager. After investigation of the problem, it was revealed that the large load was caused due to the poor configuration of the core component of the Pnfs manager, which is the namespace database. The role of the namespace database in the dCache system, is to keep all the meta data related to the files stored on the system. The feature selection procedure served as a root cause analysis approach and also it helped in further optimization of the service state modelling process.
During the case studies and in the process of the approach identification, the service state modelling has been considered as a classification problem. The neural network based approach is considered as an alternative to SVM based algorithms for the solution of the classification problems. Hence ANFIS has considered as another fundamental approach for handling the problem.
The overall accuracy of ANFIS based approach was 99.62%, but the most important components here, as well as in case of the SVM application for the classification, were the recall and the precision of the failure detection. These parameters has shown an approach sensitivity to the service failure identification and were 62.5% and 71.43% accordingly. Thus the ANFIS based approach was able to detect the most of the failures properly, which has been proven by the measurement of recall. The precision metrics has used for the identification of ratio of overall correct predictions of failures, i.e. was an indicator of the rate of false alarm caused by the approach. Hence, 71.43% of precision allowed to consider the ANFIS based approach as a reliable method for the failure detection.
The service status identification and its failure root cause analysis is important component for the automation of the failure detection and its elimination. However, performing

[2]Class recall for "True 0" and Class Precision for "Prediction 0".
[3]Weights varies from 0 to 1. Largest in this case means > 0.5

of the proactive management of infrastructure allows to avoid large number of the service or system failures. The key aspect of the proactive management, which is also considered as a part of Autonomic Computing, is the in advance assessment of the service and infrastructure related problems. In order to have an adequate assessment of the approaching problems regarding the service provisioning, it is important to have an adequate metrics characterizing the service availability. This information would make possible avoiding the SLA violations and the decreasing of the provided QoS.
Here, the Service Response Time is considered as a sufficient metrics for the characterizing the service dynamics and the final subsection[4] of chapter 7 is dedicated to the description of the SRT prediction case studies.
Due to the critical importance of the storage and batch systems functionality for entire infrastructure the "dCap" and the "pbs server" services were selected as objects for the case studies. The core of the SRT prediction approach was the representation of data in terms of the time series and applying and ANFIS for prediction. The summarizing results of the case studies shown in the following table (see table 8.1), which indicates the high accuracy of ANFIS and its advantage in comparison to another time series prediction approach, NARNET[5].
Thus, general assessment of ANFIS based approaches allows to consider the methods as

Methods	dCap	pbs server
	100% - MAPE	100% - MAPE
ANFIS 3MF	97.3%	90.4%
ANFIS 4MF	97.2%	**93.6%**
ANFIS 5MF	**97.7%**	84.6%
NARNET	97.0%	88.2%

Table 8.1.: ANFIS and NARNET accuracy for the SRT Prediction from six minutes to eight hours interval.

highly efficient and reliable for the service status identification and prediction. The main limitation of the proposed approach is the need of training procedures and provisioning of the adequate input data.
Another limitation of an approach is, the feature selection process. It is important to adequately select the most important monitoring metrics, which also can serve as a hint for the reasons for the system's poor performance. Without adequate knowledge of the infrastructure, it is hard to use the information about the ranked features for the system's detailed analysis. Thus, the approach efficiency depends on the expert's involvement, i.e. it is impossible to have the full automation using ANFIS and SVM based feature selection technique. The key aspect also goes on the need for the re-training of ANFIS. The infrastructure is affected by the constant updates and changes, hence the system behaviour cannot be modelled using the same ANFIS model, which was trained

[4]Subsection 7.2
[5]Nonlinear Autoregressive Neural Networks.

on the monitoring data from the past several years. Thus it is very important to have the adequate input data containing the system behaviour after each significant hardware and software change.
As it was shown in the service status detection case study, ANFIS is not extremely sensitive to the unbalanced data, however provisioning of sufficient system failure information can increase the method's recall and precision significantly.
However, taking into account the results of the described approach for the system status identification and prediction, the ANFIS based approach can be considered as sufficient and good candidate to be integrated in the system administration process analysis, management and automation.

8.3. Outlook

The ANFIS based approaches provide the possibility to build the models for the service status identification and its state prediction. The approach can be integrated with automatic system management tools like CFengine, which is used for the GoeGrid facility. In order to make the ANFIS applicable it is important to perform four general steps. At the first step the monitoring data needs to be aggregated, which includes the data from the monitoring tools, system log files and the SRT measurements. The next step requires the processing of the aggregated data, this step is also called the data pre-processing, which is related to the data normalization, filtering and synchronization, but it is not required for the SRT data. In case of the SRT information, it is sufficient to arrange the input and output data properly[6]. The further step for the service status identification modelling is the feature selection, which also provides the information about the most important monitoring attributes influencing the service state. The most important attributes as input and the service status as output are supplied to ANFIS as training data. The result of the training processes in case of the service status identification and its prediction are the models. The derived models are able to identify and predict the service status and state automatically, which can be used as a trigger information for certain actions in the scope of the system's proactive management. Thus, ANFIS based approach integration in the automated management tools can be considered as a next step in the optimization and efficient monitoring of the large scale computing facility. The general scheme of ANFIS based approaches integration into the system management and automation tools is depicted in the following diagram (see fig.8.1)
The SRT prediction method can be used not only for the in advance service state identification, but it can be easily extended for the prediction of other monitoring metrics like CPU load or memory consumption, as well as for the network load. Thus the approach can be considered as general method for in advance assessment of various system attributes. Future work in this direction can significantly improve and facilitate the proactive management techniques development.

[6]Using the structure of l vector.

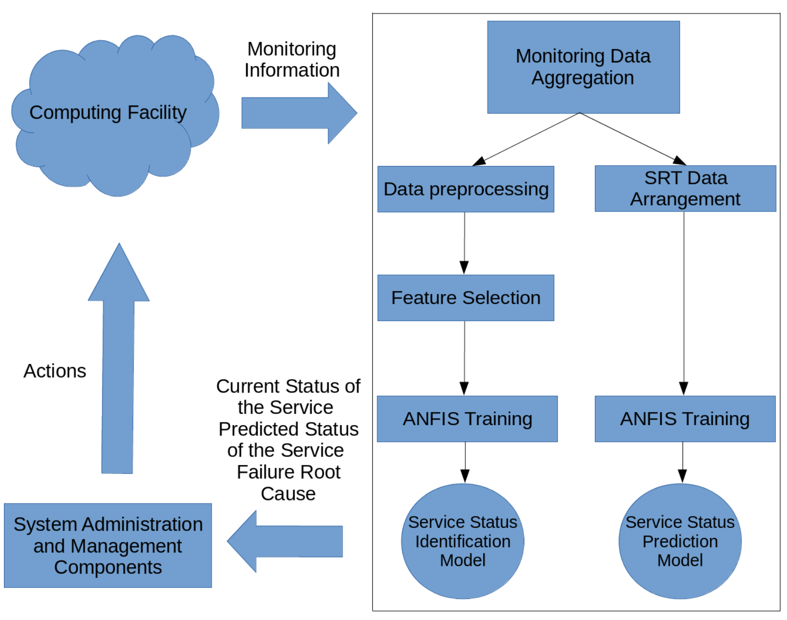

Computing Facility
Monitoring Information
Monitoring Data Aggregation
Data preprocessing
SRT Data Arrangement
Feature Selection
ANFIS Training
ANFIS Training
Service Status Identification Model
Service Status Prediction Model
Actions
Current Status of the Service Predicted Status of the Service Failure Root Cause
System Administration and Management Components

Figure 8.1.: Diagram of ANFIS based approaches integration in the large scale computing system administration and management.

9 Acknowledgments

First of all I would like to express my cordial gratitude to Prof. Dr. Arnulf Quadt for giving the chance to start my doctoral studies at the II Institute of Physics. I would like to thank him for support and guidance throughout the years, for giving the opportunities to attend a number of summer schools and workshops, as well as the conferences, where I had chance to present my research results. I would like to thank him for giving me the freedom and time for my research. Also I would like express my gratitude to Prof. Dr. Ramin Yahyapour, who agreed to be the second referee of this work.
I would like to thank Prof. Dr. Stan Lai, Prof. Dr. Jens Grabowski, Jun.-Prof. Dr. Steffen Schumann, and PD Dr. Ralf Bernhard for their willingness to evaluate the work presented here.
Without the help and support of Ms Christa Wohlfahrt, Ms Bernadette Tyson, Ms Heidi Afshar, Heike Ahrens and Ms Gabriela Herbold it would be impossible to manage number of issues very quickly and efficiently.
I want to thank all my colleges and friends at the II Institute of Physics for their support. Especially I would like to express my gratitude to Lucie Hamdi, Jens Weingarten, Jörn Grosse-Knetter, Matthias George, Leonid Serkin and Haykuhi Musheghyan.
I would like to express my special gratitude to Dr. Jörg Meyer, Dr. Boris Lemmer, Dr. Pavel Weber, Dr. Jordi Nadal and Dr.Gen Kawamura. I was lucky since I met these people. Without their advices, willingness to discuss my research and comment my work it would be impossible approaching the finish line.
I am very grateful to my family for their support. Their constant encouragement and motivation was very helpful in a every single moment of time, I've spent for my research. Finally I would like to thank my wife Mariam and my daughter Elene for their endless love and support.

Bibliography

[1] *WLCG Community, WLCG Web Page*, http://lcg.web.cern.ch/LCG, Last accessed on April 3, 2015.

[2] A. Collaboration *et al.*, *ATLAS level-1 trigger: Technical Design Report*, ATLASTDR-012, CERN-LHCC-98-014, CERN, Geneva (1998).

[3] P. H. Salus, *Casting the Net: From ARPANET to Internet and Beyond*, Addison-Wesley Longman Publishing Co., Boston, MA, USA, (1995).

[4] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, *A Brief History of the Internet*, SIGCOMM Comput. Commun. Rev. **39(5)** (Oct. 2009) 22.

[5] J. F. Shoch and J. A. Hupp, *The &Ldquo;Worm&Rdquo; Programs&Mdash;Early Experience with a Distributed Computation*, Commun. ACM **25(3)** (Mar. 1982) 172.

[6] R. Stevens, P. Woodward, T. DeFanti, and C. Catlett, *From the I-WAY to the National Technology Grid*, Commun. ACM **40(11)** (Nov. 1997) 50.

[7] Z. Zhang, C. Wu, and D. W. Cheung, *A Survey on Cloud Interoperability: Taxonomies, Standards, and Practice*, SIGMETRICS Perform. Eval. Rev. **40(4)** (Apr. 2013) 13.

[8] J. A. R. Nt, J. M. de Souza, G. Zimbrão, G. Xexéo, E. Neves, and W. A. Pinheiro, *A P2P Approach for Business Process Modelling and Reuse*, in: *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, Berlin, Heidelberg, (2006), Springer-Verlag, ISBN 3-540-38444-8, 978-3-540-38444-1, 2006 297–307.

[9] I. Al-Azzoni and D. G. Down, *Dynamic Scheduling for Heterogeneous Desktop Grids*, in: *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, GRID '08, Washington, DC, USA, (2008), IEEE Computer Society, ISBN 978-1-4244-2578-5, 2008 136–143.

[10] I. Foster, J. Geisler, B. Nickless, W. Smith, and S. Tuecke, *Software Infrastructure for the I-WAY High-performance Distributed Computing Experiment*, in: *Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing*, HPDC '96, Washington, DC, USA, (1996), IEEE Computer Society, ISBN 0-8186-7582-9, 1996 562–.

[11] F. Gagliardi, B. Jones, M. Reale, and S. Burke, *European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-science Applications*, in: M. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, vol. 2459 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, ISBN 978-3-540-44252-3, 2002 480–499.

[12] P. Avery and I. Foster, *The GriPhyN Project: Towards Petascale Virtual Data Grids, 2001.*

[13] M. Lamanna, *The LHC computing grid project at CERN*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **534(1)** (2004) 1.

[14] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*, Elsevier, (2003).

[15] I. Foster, C. Kesselman, and S. Tuecke, *The anatomy of the grid: Enabling scalable virtual organizations*, International journal of high performance computing applications **15(3)** (2001) 200.

[16] I. Foster, *What is the grid? A three point Checklist.*, GRIDToday Now: HPC in the Cloud (2002), http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf.

[17] M. Baker, R. Buyya, and D. Laforenza, *The Grid: International efforts in global computing*, Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (2000).

[18] W. E. Moen, *Realizing the information future: The internet and beyond: NRENAISSANCE COMMITTEE, NATIONAL RESEARCH COUNCIL. National Academy Press, Washington, DC (1994). 301 pp., ISBN 0-309-05044-8*, Pergamon, (1996).

[19] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, *Internet X.509 Public Key Infrastructure Proxy Certificate Profile*, Internet Engineering Task Force, 2004 .

[20] W. Allcock, *GridFTP: Protocol Extensions to FTP for the Grid*, Global Grid Forum, 2003 .

[21] J. Gu, A. Sim, and A. Shoshani, *The Storage Resource manager Interface Specification version 2.1*, 2003 `http://sdm.lbl.gov/srm/documents/joint.docs/SRM.spec.v2.1.final.doc`.

[22] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, and J. Navarro, *GLUE Specification v. 2.0*, Global Grid Forum Recommendation GFD-R-P.147, 2009 .

[23] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, *The physiology of the grid*, Grid computing: making the global infrastructure a reality (2003) 217.

[24] D. R. Simpson, N. Kelly, P. Jithesh, P. Donachy, T. Harmer, R. Perrott, J. Johnston, P. Kerr, M. McCurley, and S. McKee, *GeneGrid: A practical workflow implementation for a grid based virtual bioinformatics laboratory*, in: *Proc. of the UK e-Science All Hands Meeting*, vol. 2004, (2004), 2004 547–554.

[25] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, *The WS-Resource Framework Version 1.0 (2004)*, Initial draft release from **3(05)** (2004).

[26] J. Yu and R. Buyya, *A Taxonomy of Workflow Management Systems for Grid Computing*, Journal of Grid Computing **3(3-4)** (2005) 171.

[27] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible markup language (XML)*, World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210 (1998) 16.

[28] M. Diaz, *Petri nets: fundamental models, verification and applications*, John Wiley & Sons, (2013).

[29] A. Evans, R. B. France, K. Lano, and B. Rumpe, *Developing the UML as a Formal Modelling Notation*, CoRR **abs/1409.6928** (2014).

[30] B. Adelberg, H. Garcia-Molina, and B. Kao, *Emulating Soft Real-Time Scheduling Using Traditional Operating System Schedulers*, in: *IEEE Real-Time Systems Symposium.*, 1994 .

[31] S. Fitzgerald, I. Foster, C. Kesselman, G. V. Laszewski, W. Smith, and S. Tuecke., *A Directory Service for Configuring High-Performance Distributed Computations.*, In 6th IEEE Symposium on High-Performance Distributed Computing Proceedings (1997) 365.

[32] R. Wolski, N. T. Spring, and J. Hayes, *The network weather service: a distributed resource performance forecasting service for metacomputing*, Future Generation Computer Systems **15(5)** (1999) 757.

[33] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, *Evaluation of job-scheduling strategies for grid computing*, in: *Grid ComputingGRID 2000*, Springer, 2000 191–202.

[34] J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd, *Gridflow: Workflow management for grid computing*, in: *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on.* IEEE, (2003), 2003 198–205.

[35] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman, *Workflow management in GriPhyN*, in: *Grid Resource Management*, Springer, 2004 99–116.

[36] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny, *Pegasus: Mapping scientific workflows onto the grid*, in: *Grid Computing.* Springer, (2004), 2004 11–20.

[37] F. Berman, H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta, W. Deng, J. Dongarra, L. Johnsson, K. Kennedy, *et al.*, *New grid scheduling and rescheduling methods in the GrADS project*, International Journal of Parallel Programming **33(2-3)** (2005) 209.

[38] R. Prodan and T. Fahringer, *Dynamic scheduling of scientific workflow applications on the grid: a case study*, in: *Proceedings of the 2005 ACM symposium on Applied computing.* ACM, (2005), 2005 687–694.

[39] R. Buyya, D. Abramson, and J. Giddy, *Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid*, in: *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, vol. 1. IEEE, (2000), 2000 283–289.

[40] S. Venugopal, R. Buyya, and L. Winton, *A grid service broker for scheduling distributed data-oriented applications on global grids*, in: *Proceedings of the 2nd workshop on Middleware for grid computing.* ACM, (2004), 2004 75–80.

[41] P. A. Dinda, *Online prediction of the running time of tasks*, Cluster Computing **5(3)** (2002) 225.

[42] G. Zheng, T. Wilmarth, P. Jagadishprasad, and L. V. Kalé, *Simulation-based performance prediction for large parallel machines*, International Journal of Parallel Programming **33(2-3)** (2005) 183.

[43] G. R. Nudd, D. J. Kerbyson, E. Papaefstathiou, S. C. Perry, J. S. Harper, and D. V. Wilcox, *PACEA toolset for the performance prediction of parallel and distributed systems*, International Journal of High Performance Computing Applications **14(3)** (2000) 228.

[44] H. J. Dail, *A modular framework for adaptive scheduling in grid application development environments*, Ph.D. thesis, University of California, San Diego, (2002).

[45] S. Jang, X. Wu, V. Taylor, G. Mehta, K. Vahi, and E. Deelman, *Using performance prediction to allocate grid resources*, Texas A&M University, College Station, TX, GriPhyN Technical Report **25** (2004).

[46] A. Mayer, S. McGough, N. Furmento, W. Lee, S. Newhouse, and J. Darlington, *ICENI dataflow and workflow: Composition and scheduling in space and time*, in: *UK e-Science All Hands Meeting*, vol. 634. Citeseer, (2003), 2003 627.

[47] W. Smith, I. Foster, and V. Taylor, *Predicting application run times using historical information*, in: *Job Scheduling Strategies for Parallel Processing.* Springer, (1998), 1998 122–142.

[48] R. Buyya, D. Abramson, and J. Giddy, *A case for economy grid architecture for service oriented grid computing*, in: *Parallel and Distributed Processing Symposium, International*, vol. 2. IEEE Computer Society, (2001), 2001 20083a–20083a.

[49] A. Galstyan, K. Czajkowski, and K. Lerman, *Resource allocation in the grid using reinforcement learning*, in: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3.* IEEE Computer Society, (2004), 2004 1314–1315.

[50] D. A. Bacigalupo, S. A. Jarvis, L. He, and G. R. Nudd, *An investigation into the application of different performance prediction techniques to e-commerce applications*, in: *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International.* IEEE, (2004), 2004 248.

[51] S. Hwang and C. Kesselman, *Grid workflow: a flexible failure handling framework for the grid*, in: *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on.* IEEE, (2003), 2003 126–137.

[52] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, *Taverna: a tool for building and running workflows of services*, Nucleic acids research **34(suppl 2)** (2006) W729.

[53] G. Malewicz, I. Foster, A. L. Rosenberg, and M. Wilde, *A tool for prioritizing DAGMan jobs and its evaluation*, Journal of Grid Computing **5(2)** (2007) 197.

[54] J. H. Abawajy, *Fault-tolerant scheduling policy for grid computing systems*, in: *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International.* IEEE, (2004), 2004 238.

[55] K. Krauter, R. Buyya, and M. Maheswaran, *A taxonomy and survey of grid resource management systems for distributed computing*, Software: Practice and Experience **32(2)** (2002) 135.

[56] *Adaptive Computing, TORQUE Resource Manager*, http://docs.adaptivecomputing.com/8-1-0/basic/help.htm#topics/torque/0-intro/introduction.htm, Last accessed on March 28, 2015.

[57] *NASA Ames Research Center and Lawrence Livermore National Laboratory and Veridian Information Solutions Inc., OpenPBS Public Web Page*, http://www.mcs.anl.gov/research/projects/openpbs, Last accessed on April 3, 2015.

[58] *ORACLE, Oracle Grid Engine EGEE*, http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html, Last accessed on April 29, 2015.

[59] *IBM, Workload Management with LoadLeveler*, http://www.redbooks.ibm.com/abstracts/sg246038.html, Last accessed on July 2, 2015.

[60] *Microsoft, Microsoft HPC Pack (Windows HPC Server)*, https://technet.microsoft.com/en-us/library/cc514029.aspx, Last accessed on June 2, 2015.

[61] *Globus and Globus Alliance and Globus Toolkit, Globus Toolkit 6*, http://toolkit.globus.org/toolkit/docs/6.0, Last accessed on March 20, 2015.

[62] *Julich Supercomputing Centre, Unicore*, https://www.unicore.eu/documentation, Last accessed on April 5, 2015.

[63] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman, *A metadata catalog service for data intensive applications*, in: *Supercomputing, 2003 ACM/IEEE Conference*. IEEE, (2003), 2003 33–33.

[64] I. Foster, A. Iamnitchi, M. Ripeanu, A. Chervenak, E. Deelman, C. Kesselman, W. Hoschek, P. Kunszt, H. Stockinger, K. Stockinger, *et al.*, *Giggle: A framework for constructing scalable replica location services*, in: *Conference On High Performance Networking and Computing*. Citeseer, (2002), 2002 .

[65] K. Youhei Morita and S. Noriyuki Soda, *GFARM V2: A GRID FILE SYSTEM THAT SUPPORTS HIGH-PERFORMANCE DISTRIBUTED AND PARALLEL DATA COMPUTING*.

[66] P. Honeyman, W. A. Adamson, and S. McKee, *GridNFS: global storage for global collaborations*, in: *Local to Global Data Interoperability-Challenges and Technologies, 2005*. IEEE, (2005), 2005 111–115.

[67] M. Branco, *Don Quijote-Data Management for the ATLAS Automatic Production System*.

[68] F. T. Protocol, J. Postel, and J. Reynolds, *File Transfer Protocol*, Tech. rep., Internet RFC 959, October, (1985).

[69] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext transfer protocol–HTTP/1.1*.

[70] *UberFTP Dynamic Client of the GridFTP Protocol*, http://toolkit.globus.org/toolkit/docs/3.2/gridftp/user/interactiveclient.html, Last accessed on March 27, 2015.

[71] *The Chirp I/O Protocol*, http://research.cs.wisc.edu/htcondor/chirp/, Last accessed on March 27, 2015.

[72] *Data Link Client Access Protocol (DCAP) Specifications*, http://www.networksorcery.com/enp/protocol/dcap.htm, Last accessed on March 27, 2015.

[73] G. Kola and M. Livny, *Diskrouter: A flexible infrastructure for high performance large scale data transfers*, UW–Madison, Tech. Rep. CS-TR-2004-1518 (2003).

[74] A. Bassi, M. Beck, T. Moore, J. S. Plank, M. Swany, R. Wolski, and G. Fagg, *The Internet Backplane Protocol: A study in resource sharing*, Future Generation Computer Systems **19(4)** (2003) 551.

[75] M. Beck and T. Moore, *Logistical networking: a global storage network*, in: *Journal of Physics: Conference Series*, vol. 16. IOP Publishing, (2005), 2005 531.

[76] J. S. Plank, M. Beck, W. R. Elwasif, T. Moore, M. Swany, and R. Wolski, *The internet backplane protocol: Storage in the network*, in: *In Proceedings of the Network Storage Symposium*. Citeseer, (1999), 1999 .

[77] D. Thain, J. Basney, S.-C. Son, and M. Livny, *The kangaroo approach to data movement on the grid*, in: *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*. IEEE, (2001), 2001 325–333.

[78] J. Bent, V. Venkataramani, N. LeRoy, A. Roy, J. Stanley, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, *Flexibility, manageability, and performance in a grid storage appliance*, in: *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*. IEEE, (2002), 2002 3–12.

[79] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, *Grid information services for distributed resource sharing*, in: *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*. IEEE, (2001), 2001 181–194.

[80] W. Yeong, T. Howes, and S. Kille, *Lightweight directory access protocol*.

[81] Z. Balaton, G. Gombás, and Z. Németh, *A Novel architecture for grid information systems*, in: *Cluster Computing and the Grid, IEEE International Symposium on*. IEEE Computer Society, (2002), 2002 274–274.

[82] J. M. Schopf, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D'Arcy, and A. Chervenak, *Monitoring the grid with the Globus Toolkit MDS4*, in: *Journal of Physics: Conference Series*, vol. 46. IOP Publishing, (2006), 2006 521.

[83] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, *Modeling and managing state in distributed systems: The role of OGSI and WSRF*, Proceedings of the IEEE **93(3)** (2005) 604.

[84] *OGF (Open Grid Forum)*, http://www.ogf.org/, Last accessed on March 28, 2015.

[85] *LCG (LHC Computing Grid)*, http://www.cern.ch/lcg, Last accessed on June 8, 2015.

[86] *OSG (Open Science Grid)*, http://www.opensciencegrid.org/, Last accessed on March 15, 2015.

[87] *NDGF (Nordic DataGrid Facility)*, http://www.ndgf.org/, Last accessed on April 15, 2015.

[88] *EGEE (Enabling Grids for E-sciencE)*, http://www.eu-egee.org/, Last accessed on March 28, 2015.

[89] *EGI (European Grid Infrastructure)*, http://www.egi.eu/, Last accessed on March 28, 2015.

[90] S. Burke, L. Field, and D. Horat, *Migration to the GLUE 2.0 information schema in the LCG/EGEE/EGI production Grid*, in: *Journal of Physics: Conference Series*, vol. 331. IOP Publishing, (2011), 2011 062004.

[91] *Berkeley Database Information Index (BDII)*, http://twiki.cern.ch/twiki/bin/view/EGEE/Glite-BDII, Last accessed on April 15, 2015.

[92] *UNICORE Common Information Service (CIS)*, http://www.unicore.eu/documentation/manuals/unicore6/files/cis/cis-manual.html, Last accessed on March 15, 2015.

[93] *Grid Information System*, http://gridinfo.web.cern.ch/, Last accessed on March 15, 2015.

[94] *Models of Networked Analysis at Regional Centres for LHC Experiments (MONARC)*, http://monarc.web.cern.ch/MONARC/, Last accessed on June 8, 2015.

[95] A. collaboration *et al.*, *ALICE Technical Design Report of the photon spectrometer (PHOS)*, CERN/LHCC (1999) 99.

[96] W. Armstrong, W. Burris, D. Gingrich, P. Green, L. Greeniaus, J. Hewlett, L. Holm, J. McDonald, S. Mullin, W. Olsen, *et al.*, *ATLAS: technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN.*

[97] C. collaboration *et al.*, *CMS technical proposal*, CERN/LHCC **94** (1994) 38.

[98] L. Collaboration *et al.*, *LHCb technical proposal*, CERN/LHCC **4** (1998) 1998.

[99] *European Organization for Nuclear Research (CERN)*, http://public.web.cern.ch/public/, Last accessed on March 28, 2015.

[100] *Grid Configuration Database (GOCDB)*, http://goc.egi.eu/, Last accessed on March 28, 2015.

[101] *Operations Portal (CIC)*, http://operations-portal.egi.eu/, Last accessed on March 29, 2015.

[102] *Global Grid User Support EGEE*, http://ggus.eu/, Last accessed on March 28, 2015.

[103] R. Byrom, R. Cordenonsi, L. Cornwall, M. Craig, A. Djaoui, A. Duncan, S. Fisher, J. Gordon, S. Hicks, D. Kant, *et al.*, *APEL: An implementation of Grid accounting using R-GMA*, in: *UK e-Science All Hands Conference*, (2005), 2005 .

[104] A. Duarte, P. Nyczyk, A. Retico, and D. Vicinanza, *Monitoring the EGEE/WLCG grid services*, in: *Journal of Physics: Conference Series*, vol. 119. IOP Publishing, (2008), 2008 052014.

[105] J. A. et al., *Dashboard applications to monitor experiment activities at sites*, Journal of Physics: Conference Series 219 (2010) 062003 (2010).

[106] C. Aiftimiei, A. Aimar, A. Ceccanti, M. Cecchi, A. Di Meglio, F. Estrella, P. Fuhrmam, E. Giorgio, B. Konya, L. Field, *et al.*, *Towards next generations of software for distributed infrastructures: the European Middleware Initiative*, in: *E-Science (e-Science), 2012 IEEE 8th International Conference on*. IEEE, (2012), 2012 1–10.

[107] M. Grønager *et al.*, *LCG and ARC middleware interoperability*, Proceedings of Computing in High Energy Physics (CHEP 2006), Mumbai, India (2006).

[108] S. Campana, A. Brown, D. Bonacorsi, V. Capone, D. De Girolamo, A. Casani, J. Flix, A. Forti, I. Gable, O. Gutsche, *et al.*, *Deployment of a wlcg network monitoring infrastructure based on the perfsonar-ps technology*, in: *Journal of Physics: Conference Series*, vol. 513. IOP Publishing, (2014), 2014 062008.

[109] *Amazon Elastic Computing Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/, Last accessed on March 16, 2015.

[110] *Google Cloud Platform*, http://cloud.google.com, Last accessed on March 15, 2015.

[111] *Oak Ridge Leadership Computing Facility (OLCF)*, http://http://www.olcf.ornl.gov, Last accessed on March 28, 2015.

[112] *Future Grid (Future Systems)*, http://portal.futuresystems.org, Last accessed on March 15, 2015.

[113] *Argonne Leadership Computing Facility (MIRA supercomputer) Amazon Elastic Computing Cloud (Amazon EC2)*, http://www.alcf.anl.gov/mira, Last accessed on March 16, 2015.

[114] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, *Condor-G: A computation management agent for multi-institutional grids*, Cluster Computing **5(3)** (2002) 237.

[115] A. Filipčič *et al.*, *arcControlTower: the System for Atlas Production and Analysis on ARC*, in: *Journal of Physics: Conference Series*, vol. 331. IOP Publishing, (2011), 2011 072013.

[116] P. Nilsson, J. Caballero, K. De, T. Maeno, A. Stradling, T. Wenaus, A. Collaboration, *et al.*, *The ATLAS PanDA Pilot in Operation*, in: *Journal of Physics: Conference Series*, vol. 331. IOP Publishing, (2011), 2011 062040.

[117] R. Jones and D. Barberis, *The ATLAS computing model*, in: *Journal of Physics: Conference Series*, vol. 119. IOP Publishing, (2008), 2008 072020.

[118] R. Jones and D. Barberis, *The evolution of the ATLAS computing model*, in: *Journal of Physics: Conference Series*, vol. 219. IOP Publishing, (2010), 2010 072037.

[119] V. Garonne, G. A. Stewart, M. Lassnig, A. Molfetas, M. Barisits, T. Beermann, A. Nairz, L. Goossens, F. B. Megino, C. Serfon, *et al.*, *The atlas distributed data management project: Past and future*, in: *Journal of Physics: Conference Series*, vol. 396. IOP Publishing, (2012), 2012 032045.

[120] V. Garonne, R. Vigne, G. Stewart, M. Barisits, M. Lassnig, C. Serfon, L. Goossens, A. Nairz, A. Collaboration, *et al.*, *Rucio–The next generation of large scale distributed system for ATLAS Data Management*, in: *Journal of Physics: Conference Series*, vol. 513. IOP Publishing, (2014), 2014 042021.

[121] B. Leiba, *Oauth web authorization protocol*, IEEE Internet Computing **16(1)** (2012) 74.

[122] A. Anisenkov, S. Belov, A. Di Girolamo, S. Gayazov, A. Klimentov, D. Oleynik, and A. Senchenko, *AGIS: The ATLAS Grid Information System*, in: *Journal of Physics: Conference Series*, vol. 396. IOP Publishing, (2012), 2012 032006.

[123] *The Django Web Framework*, http://www.djangoproject.com/, Last accessed on June 15, 2015.

[124] J. Elmsheuser, F. Hönig, F. Legger, R. M. Llamas, F. Sciacca, D. van der Ster, *et al.*, *Grid site testing for ATLAS with HammerCloud*, vol. 513. IOP Publishing, (2014), 2014 032030.

[125] *ATLAS DDM Dashboard*, http://dashb-atlas-ddm-acc.cern.ch/dashboard/request.py/ddmaccounting, Last accessed on April 15, 2015.

[126] *GoeGrid Computing Centre*, http://goegrid.gwdg.de, Last accessed on March 28, 2015.

[127] *II Physics Institute of Georg-August-Universität Göttingen*, http://physik2.uni-goettingen.de, Last accessed on June 8, 2015.

[128] *Institute for Theoretical Physics of Georg-August-Universität Göttingen*, http://www.uni-goettingen.de/en/200460.html, Last accessed on June 8, 2015.

[129] *MediGRID*, http://www.medigrid.de, Last accessed on June 8, 2015.

[130] *TextGrid*, http://www.textgrid.de, Last accessed on June 8, 2015.

[131] *Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen - GWDG*, http://www.gwdg.de, Last accessed on March 28, 2015.

[132] *CentOS Linux Home Page*, http://centos.org/, Last accessed on April 15, 2015.

[133] *Storage Management System dCache*, http://www.dcache.org, Last accessed on June 15, 2015.

[134] The ATLAS Collaboration, G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003.

[135] *TORQUE Resource Manager*, http://www.adaptivecomputing.com/products/open-source/torque/, Last accessed on March 20, 2015.

[136] *MAUI Computational Task Scheduler*, http://www.adaptivecomputing.com/products/open-source/maui/, Last accessed on March 28, 2015.

[137] D. J. Quinn, D. Jackson, Q. Snell, and M. Clement, *Core Algorithms of the Maui Scheduler*.

[138] *Deutsches Elektronen-Synchrotron, Ein Forschungszentrum der Helmholtz-Gemeinschaft, Hamburg*, http://www.desy.de/, Last accessed on March 27, 2015.

[139] *Fermi National Accelerator Laboratory, U.S. Department of Energy, Managed by Fermi Research Alliance, LLC, Batavia, Illinois, USA*, http://www.fnal.gov/, Last accessed on March 28, 2015.

[140] *The Use of PNFS in dCache*, http://www.dcache.org/manuals/Book-1.9.5/config/cf-pnfs-use.shtml, Last accessed on March 28, 2015.

[141] E. J. Whitehead Jr and Y. Y. Goland, *WebDav*, in: *ECSCW99.* Springer, (2002), 2002 291–310.

[142] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, *XROOTD-A Highly scalable architecture for data access*, WSEAS Transactions on Computers **1(4.3)** (2005).

[143] R. Perrey and M. Lycett, *Service-oriented architecture*, in: *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on.* IEEE, (2003), 2003 116–119.

[144] P. Andreetto, S. Bertocco, F. Capannini, M. Cecchi, A. Dorigo, E. Frizziero, A. Gianelle, F. Giacomini, M. Mezzadri, S. Monforte, *et al.*, *Status and Developments of the CREAM Computing Element Service*, in: *Journal of Physics: Conference Series*, vol. 331. IOP Publishing, (2011), 2011 062024.

[145] U. Egede, K. Harrison, R. Jones, A. Maier, J. Moscicki, G. Patrick, A. Soroko, and C. Tan, *Ganga user interface for job definition and management*, in: *Proc. 4th UK e-Science All-Hands Meeting, Nottingham, UK*, (2005), 2005 .

[146] P. Nilsson, J. Caballero, K. De, T. Maeno, M. Potekhin, and T. Wenaus, *The PanDA system in the Atlas experiment*, in: *ACAT08 Conference*, (2008), 2008 .

[147] E. Krepska, T. Kielmann, R. Sirvent, and R. M. Badia, *A service for reliable execution of grid applications*, in: *Achievements in European Research on Grid Systems*, Springer, 2008 179–192.

[148] E. Molinari, F. Prelz, D. Rebatto, *et al.*, *A local batch system abstraction layer for global use*, in: *Proc. of XV International Conference on Computing in High Energy and Nuclear Physics (CHEP 2006)*, (2006), 2006 .

[149] M. Sgaravatto, *CREAM Job Description Language Attributes Specification for the EGEE Middleware document Identifier EGEE-JRA1-TEC-592336.*

[150] *Rutherford Appleton Laboratory (RAL)*, http://www.stfc.ac.uk/, Last accessed on March 25, 2015.

[151] I. Habib, *Virtualization with kvm*, Linux Journal **2008(166)** (2008) 8.

[152] *Open Source Toolkit for Real and Virtual Cluster - ROCKS cluster tools*, http://www.rocksclusters.org/, Last accessed on March 29, 2015.

[153] *Rocks Cluster Distribution: Users Guide*, http://www.rocksclusters.org/rocks-documentation/4.2/introduction.html, Last accessed on March 28, 2015.

[154] *RedHat KICKSTART INSTALLATIONS*, https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/ch-kickstart2.html, Last accessed on March 28, 2015.

[155] *Ganglia Monitoring Tool*, http://ganglia.sourceforge.net/, Last accessed on June 28, 2015.

[156] D. Zamboni, *Learning CFEngine 3: Automated system administration for sites of any size*, " O'Reilly Media, Inc.", (2012).

[157] *Nagios Monitoring Tool*, http://www.nagios.org/, Last accessed on March 28, 2015.

[158] W. Barth, *Nagios: System and network monitoring*, No Starch Press, (2008).

[159] *Ganglia monitoring tool technical guide*, http://github.com/ganglia/monitor-core/wiki/Ganglia-Quick-Start, Last accessed on May 28, 2015.

[160] V. Mauch, C. Ay, S. Birkholz, V. Büge, A. Burgmeier, J. Meyer, F. Nowak, A. Quadt, G. Quast, P. Sauerland, *et al.*, *The HappyFace project*, in: *Journal of Physics: Conference Series*, vol. 331. IOP Publishing, (2011), 2011 082011.

[161] S. Zanikolas and R. Sakellariou, *A taxonomy of grid monitoring systems*, Future Generation Computer Systems **21(1)** (2005) 163.

[162] *Python Programming Language Official Website*, http://www.python.org, Last accessed on July 15, 2015.

[163] *CherryPy - A Minimalist Python Web Framework*, http://www.cherrypy.org/, Last accessed on March 16, 2015.

[164] *HappyFace Instance at ATLAS Distributed Computing Tier 2 Centre GoeGrid*, http://happyface-goegrid.gwdg.de, Last accessed on June 9, 2015.

[165] *Service Availability Monitoring for ATLAS Distributed Computing Tier 2 Centre GoeGrid*, http://wlcg-sam-atlas.cern.ch/templates/ember/#/plot?group=All%20sites&profile=ATLAS_CRITICAL&sites=GoeGrid, Last accessed on June 9, 2015.

[166] E. Mainsah, *Autonomic computing: the next era of computing*, Electronics & Communication Engineering Journal **14(1)** (2002) 2.

[167] *Defense Advanced Research Projects Agency (DARPA)*, http://www.darpa.mil/, Last accessed on March 27, 2015.

[168] H. S. Kenyon, *Battlefield cognizance tool points to future*, SIGNAL Mag (2001).

[169] *DASADA project*, http://isr.uci.edu/projects/dasada/, Last accessed on April 15, 2015.

[170] D. Garlan, B. Schmerl, and J. Chang, *Using gauges for architecture-based monitoring and adaptation.*

[171] J. M. Cobleigh, L. J. Osterweil, A. Wise, and B. S. Lerner, *Containment units: A hierarchically composable architecture for adaptive systems*, in: *Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering.* ACM, (2002), 2002 159–165.

[172] G. Kaiser, P. Gross, G. Kc, J. Parekh, and G. Valetto, *An approach to autonomizing legacy systems*, Tech. rep., DTIC Document, (2005).

[173] P. Horn, *Autonomic computing: IBM's Perspective on the State of Information Technology.*

[174] L. Badger, *Self-regenerative systems(SRS) program abstract*, DARPA 2004 (2004).

[175] J. O. Kephart and D. M. Chess, *The vision of autonomic computing*, Computer **36(1)** (2003) 41.

[176] A. Hochstein, R. Zarnekow, and W. Brenner, *ITIL as common practice reference model for IT service management: formal assessment and implications for practice*, in: *e-Technology, e-Commerce and e-Service, 2005. EEE'05. Proceedings. The 2005 IEEE International Conference on.* IEEE, (2005), 2005 704–710.

[177] A. Computing, *An architectural blueprint for autonomic computing*, IBM Publication (2003).

[178] P. Horn and A. Computing, *IBMs Perspective on the State of Information Technology*, IBM TJ Watson Labs, NY, 15th October (2001).

[179] W. R. Ashby, *Design for a Brain*, Springer Science & Business Media, (1960).

[180] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, and H. Liu, *The autonomic computing paradigm*, Cluster Computing **9(1)** (2006) 5.

[181] A. G. Ganek and T. A. Corbi, *The dawning of the autonomic computing era*, IBM systems Journal **42(1)** (2003) 5.

[182] I. T. I. Orchestrator, *IBM Tivoli Provisioning Manager*, Overview Guide, Version **1(1)** (2003).

[183] M. Rahman, R. Ranjan, R. Buyya, and B. Benatallah, *A taxonomy and survey on autonomic management of applications in grid computing environments*, Concurrency and computation: practice and experience **23(16)** (2011) 1990.

[184] D. Thain, T. Tannenbaum, and M. Livny, *Condor and the Grid*, Grid computing: Making the global infrastructure a reality (2003) 299.

[185] J. Basney, M. Livny, and T. Tannenbaum, *Deploying a high throughput computing cluster*, High performance cluster computing **1(5)** (1999) 356.

[186] S. J. Chapin, D. Katramatos, J. Karpovich, and A. S. Grimshaw, *The legion resource management system*, in: *Job Scheduling Strategies for Parallel Processing.* Springer, (1999), 1999 162–178.

[187] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, G. Glover, M. Pocock, A. Wipat, *et al.*, *A tool for the composition and enactment of bioinformatics workflows. Accepted for*, Bioinformatics **16** (2004).

[188] R. D. Stevens, A. J. Robinson, and C. A. Goble, *myGrid: personalised bioinformatics on the information grid*, Bioinformatics **19(suppl 1)** (2003) i302.

[189] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *et al.*, *Web services description language (WSDL) 1.1.*

[190] K. Jackson, *OpenStack cloud computing cookbook*, Packt Publishing Ltd, (2012).

[191] D.-W. Zhang, F.-Q. Sun, X. Cheng, and C. Liu, *Research on hadoop-based enterprise file cloud storage system*, in: *Awareness Science and Technology (iCAST), 2011 3rd International Conference on.* IEEE, (2011), 2011 434–437.

[192] K. M. Sim, *Agent-based cloud computing*, Services Computing, IEEE Transactions on **5(4)** (2012) 564.

[193] P. T. Endo, D. Sadok, and J. Kelner, *Autonomic Cloud Computing: giving intelligence to simpleton nodes*, in: *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on.* IEEE, (2011), 2011 502–505.

[194] P. Bodık, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson, *Statistical machine learning makes automatic control practical for internet datacenters*, in: *Proceedings of the 2009 conference on Hot topics in cloud computing*, (2009), 2009 12–12.

[195] M. Malathi, *Cloud computing concepts*, in: *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 6. IEEE, (2011), 2011 236–239.

[196] D. J. Dubois, *Fuzzy sets and systems: theory and applications*, vol. 144, Academic press, (1980).

[197] B. Yegnanarayana, *Artificial neural networks*, PHI Learning Pvt. Ltd., (2009).

[198] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, *Engineering self-adaptive systems through feedback loops*, in: *Software engineering for self-adaptive systems*, Springer, 2009 48–70.

[199] M. Ravindranathan and R. Leitch, *Heterogeneous intelligent control systems*, in: *Control Theory and Applications, IEE Proceedings-*, vol. 145. IET, (1998), 1998 551–558.

[200] M. Naqvi, *Claims and supporting evidence for self-adaptive systems–A literature review.*

[201] M. Salehie and L. Tahvildari, *Towards a goal-driven approach to action selection in self-adaptive software*, Software: Practice and Experience **42(2)** (2012) 211.

[202] F. D. Macías-Escrivá, R. Haber, R. del Toro, and V. Hernandez, *Self-adaptive systems: A survey of current approaches, research challenges and applications*, Expert Systems with Applications **40(18)** (2013) 7267.

[203] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, *Rainbow: Architecture-based self-adaptation with reusable infrastructure*, Computer **37(10)** (2004) 46.

[204] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif, and U. Richter, *Adaptivity and self-organization in organic computing systems*, ACM Transactions on Autonomous and Adaptive Systems (TAAS) **5(3)** (2010) 10.

[205] G. T. Heineman and W. T. Councill, *Component-based software engineering*, Putting the Pieces Together, Addison-Westley (2001).

[206] N. Bencomo, P. Grace, C. Flores, D. Hughes, and G. Blair, *Genie: Supporting the model driven development of reflective, component-based adaptive systems*, in: *Proceedings of the 30th international conference on Software engineering.* ACM, (2008), 2008 811–814.

[207] A. Mishra and A. Misra, *Component assessment and proactive model for support of dynamic integration in self adaptive system*, ACM SIGSOFT Software Engineering Notes **34(4)** (2009) 1.

[208] D. Weyns and M. Georgeff, *Self-adaptation using multiagent systems*, Software, IEEE **27(1)** (2010) 86.

[209] K. Yeom and J.-H. Park, *Morphological approach for autonomous and adaptive systems based on self-reconfigurable modular agents*, Future Generation Computer Systems **28(3)** (2012) 533.

[210] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight Jr, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss, *Amorphous computing*, Communications of the ACM **43(5)** (2000) 74.

[211] P. Leitão, *A bio-inspired solution for manufacturing control systems*, in: *Innovation in manufacturing networks*, Springer, 2008 303–314.

[212] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback control of dynamics systems*, Addison-Wesley, Reading, MA (1994).

[213] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback control of computing systems*, John Wiley & Sons, (2004).

[214] K. Geihs, P. Barone, F. Eliassen, J. Floch, R. Fricke, E. Gjorven, S. Hallsteinsen, G. Horn, M. U. Khan, A. Mamelli, *et al.*, *A comprehensive solution for application-level adaptation*, Software: Practice and Experience **39(4)** (2009) 385.

[215] T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker, *Model-driven architectural monitoring and adaptation for autonomic systems*, in: *Proceedings of the 6th international conference on Autonomic computing.* ACM, (2009), 2009 67–68.

[216] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, *A survey on engineering approaches for self-adaptive systems*, Pervasive and Mobile Computing **17** (2015) 184.

[217] F. Salfner, M. Lenk, and M. Malek, *A survey of online failure prediction methods*, ACM Computing Surveys (CSUR) **42(3)** (2010) 10.

[218] H. R. Berenji, J. Ametha, and D. Vengerov, *Inductive learning for fault diagnosis*, in: *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, vol. 1. IEEE, (2003), 2003 726–731.

[219] D. W. Ho, P.-A. Zhang, and J. Xu, *Fuzzy wavelet networks for function learning*, Fuzzy Systems, IEEE Transactions on **9(1)** (2001) 200.

[220] M. H. Ning, Q. Yong, H. Di, C. Ying, and Z. J. Zhong, *Software aging prediction model based on fuzzy wavelet network with adaptive genetic algorithm*, in: *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on.* IEEE, (2006), 2006 659–666.

[221] J. L. Hellerstein, F. Zhang, and P. Shahabuddin, *An approach to predictive detection for service management*, in: *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on.* IEEE, (1999), 1999 309–322.

[222] R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, and S. M. Weiss, *Predictive algorithms in the management of computer systems*, IBM Systems Journal **41(3)** (2002) 461.

[223] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, *Critical event prediction for proactive management in large-scale computer clusters*, in: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, (2003), 2003 426–435.

[224] D. Turnbull and N. Alldrin, *Failure prediction in hardware systems*, Tech. rep., Tech. rep., University of California, San Diego. available at http://www. cs. ucsd. edu/˜ dturnbul/Papers/ServerPrediction. pdf, (2003).

[225] V. Agrawal, C. Bhattacharyya, T. Niranjan, and S. Susarla, *Prediction of Hard Drive Failures via Rule Discovery from AutoSupport Data.*

[226] L. Jouffe, *Fuzzy inference system learning by reinforcement methods*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **28(3)** (1998) 338.

[227] L. A. Zadeh, *Fuzzy sets*, Information and control **8(3)** (1965) 338.

[228] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]*, Automatic Control, IEEE Transactions on **42(10)** (1997) 1482.

[229] D. Dubois and H. Prade, *A review of fuzzy set aggregation connectives*, Information sciences **36(1)** (1985) 85.

[230] M. Grabisch, M. Sugeno, and T. Murofushi, *Fuzzy measures and integrals: theory and applications*, Springer-Verlag New York, Inc., (2000).

[231] R. R. YAGER, *On the measure of fuzziness and negation part I: membership in the unit interval.*

[232] E. Cox, *The Fuzzy Systems Handbook: A Practitioners Guide to Building, Using, and Maintaining Fuzzy Systems AP Professional*, San Diego (1999).

[233] E. H. Mamdani and S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, International journal of man-machine studies **7(1)** (1975) 1.

[234] Y. Tsukamoto, *An approach to fuzzy reasoning method*, Advances in fuzzy set theory and applications **137** (1979) 149.

[235] M. Sugeno and G. Kang, *Structure identification of fuzzy model*, Fuzzy sets and systems **28(1)** (1988) 15.

[236] T. Takagi and M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control*, Systems, Man and Cybernetics, IEEE Transactions on **(1)** (1985) 116.

[237] C. M. Bishop, *Pattern recognition and machine learning*, springer, (2006).

[238] X. Yao, *Evolving artificial neural networks*, Proceedings of the IEEE **87(9)** (1999) 1423.

[239] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman, *Applied linear statistical models*, vol. 4, Irwin Chicago, (1996).

[240] M. T. Hagan and M. B. Menhaj, *Training feedforward networks with the Marquardt algorithm*, Neural Networks, IEEE Transactions on **5(6)** (1994) 989.

[241] S. S. Haykin, *Neural networks and learning machines*, vol. 3, Pearson Education Upper Saddle River, (2009).

[242] Y.-W. Chang and C.-J. Lin, *Feature ranking using linear svm*, Causation and Prediction Challenge Challenges in Machine Learning **2** (2008) 47.

[243] T. Oetiker, *Round robin database tool.*

[244] S. Konishi, *Normalizing transformations of some statistics in multivariate analysis*, Biometrika **68(3)** (1981) 647.

[245] F. Akthar and C. Hahne, *RapidMiner 5 Operator Reference*, Rapid-I GmbH (2012).

[246] J. MacQueen *et al.*, *Some methods for classification and analysis of multivariate observations*, in: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1. Oakland, CA, USA., (1967), 1967 281–297.

[247] *Documentation of the SVM Implementation in RapidMiner; Web Source of the SVM Implementation in RapidMiner*, `http://docs.rapidminer.com/studio/operators/modeling/classification_and_regression/svm/support_vector_machine.html`, Last accessed on August 12, 2015.

[248] A. Rakotomamonjy, *Variable selection using svm based criteria*, The Journal of Machine Learning Research **3** (2003) 1357.

[249] *Documentation of the Data Sampling and Balancing Implementation in RapidMiner; Web Source of the Data Sampling and Balancing Implementation in RapidMiner*, http://docs.rapidminer.com/studio/operators/data_transformation/filtering/sampling/sample_stratified.html, Last accessed on August 15, 2015.

[250] I. MathWorks and W.-c. Wang, *Fuzzy Logic Toolbox: for Use with MATLAB: User's Guide*, Mathworks, Incorporated, (1998).

[251] M. U. Guide, *The MathWorks Inc*, Natick, MA **4** (1998) 382.

[252] J. B. BIJ, R. Edu, and K. P. B. BENNEK, *Regression error characteristic curves*, in: *Twentieth International Conference on Machine Learning (ICML-2003). Washington, DC*, (2003), 2003 .

[253] J. Pucheta, *M., C. Rodríguez Rivero, M. Herrera, C. Salas, D. Patiño and B. Kuchen.A Feed-forward Neural Networks-Based Nonlinear Autoregressive Model for Forecasting Time Series*, Revista Computación y Sistemas, Centro de Investigación en Computación-IPN, México DF, México, Computación y Sistemas **14(4)** () 423.

[254] *XXIV International Symposium on Nuclear Electronics & Computing, NEC 2013*, http://nec2013.jinr.ru, Last accessed on September 1, 2015.

[255] *21st International Conference on Computing in High Energy and Nuclear Physics, CHEP 2015*, http://chep2015.kek.jp/, Last accessed on September 1, 2015.

[256] *NUCLEAR ELECTRONICS and COMPUTING, NEC 2013, Proceedings of the XXIV International Symposium*, http://nec2013.jinr.ru/pdf/nec2013.pdf, Last accessed on September 1, 2015.

Appendices

A

RapidMiner Streams of SVM Implementations

Listing A.1: SVM classification process in RapidMiner for all eighteen numeric attributes label

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="6.5.002">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="6.5.002"
      expanded="true" name="Process">
    <parameter key="logverbosity" value="init"/>
    <parameter key="random_seed" value="2001"/>
    <parameter key="send_mail" value="never"/>
    <parameter key="notification_email" value=""/>
    <parameter key="process_duration_for_mail" value="30"/>
    <parameter key="encoding" value="SYSTEM"/>
    <process expanded="true">
      <operator activated="true" class="retrieve" compatibility="
          6.5.002" expanded="true" height="60" name="Retrieve
          ZNormalizedShiftedby3UnitsGangliadCacheData" width="90" x="45"
           y="30">
        <parameter key="repository_entry" value="//Local Repository/
            Path to the Imported Data Source"/>
      </operator>
      <operator activated="true" class="x_validation" compatibility="
          5.0.000" expanded="true" height="112" name="Validation" width=
          "90" x="380" y="30">
        <parameter key="create_complete_model" value="false"/>
        <parameter key="average_performances_only" value="true"/>
        <parameter key="leave_one_out" value="false"/>
        <parameter key="number_of_validations" value="10"/>
        <parameter key="sampling_type" value="2"/>
        <parameter key="use_local_random_seed" value="false"/>
        <parameter key="local_random_seed" value="1992"/>
        <process expanded="true">
          <operator activated="true" class="support_vector_machine" compatibility="6.5.002" expanded="true" height="112" name="SVM" width="90" x="179" y="30">
            <parameter key="kernel_type" value="dot"/>
            <parameter key="kernel_gamma" value="1.0"/>
            <parameter key="kernel_sigma1" value="1.0"/>
            <parameter key="kernel_sigma2" value="0.0"/>
            <parameter key="kernel_sigma3" value="2.0"/>
            <parameter key="kernel_shift" value="1.0"/>
            <parameter key="kernel_degree" value="2.0"/>
            <parameter key="kernel_a" value="1.0"/>
            <parameter key="kernel_b" value="0.0"/>
            <parameter key="kernel_cache" value="200"/>
            <parameter key="C" value="0.0"/>
            <parameter key="convergence_epsilon" value="0.001"/>
            <parameter key="max_iterations" value="100000"/>
            <parameter key="scale" value="true"/>
            <parameter key="calculate_weights" value="true"/>
            <parameter key="return_optimization_performance" value="true"/>
            <parameter key="L_pos" value="1.0"/>
            <parameter key="L_neg" value="1.0"/>
            <parameter key="epsilon" value="0.0"/>
            <parameter key="epsilon_plus" value="0.0"/>
            <parameter key="epsilon_minus" value="0.0"/>
            <parameter key="balance_cost" value="false"/>
            <parameter key="quadratic_loss_pos" value="false"/>
            <parameter key="quadratic_loss_neg" value="false"/>
            <parameter key="estimate_performance" value="false"/>
          </operator>
          <connect from_port="training" to_op="SVM" to_port="training set"/>
          <connect from_op="SVM" from_port="model" to_port="model"/>
          <connect from_op="SVM" from_port="weights" to_port="through 1"/>
          <portSpacing port="source_training" spacing="0"/>
          <portSpacing port="sink_model" spacing="0"/>
          <portSpacing port="sink_through 1" spacing="0"/>
          <portSpacing port="sink_through 2" spacing="0"/>
        </process>
        <process expanded="true">
          <operator activated="true" class="apply_model" compatibility="5.0.000" expanded="true" height="76" name="Apply Model" width="90" x="45" y="30">
            <list key="application_parameters"/>
            <parameter key="create_view" value="false"/>
          </operator>
          <operator activated="true" class="performance" compatibility="5.0.000" expanded="true" height="76" name="Performance" width="90" x="179" y="30">
            <parameter key="use_example_weights" value="true"/>
          </operator>
          <connect from_port="model" to_op="Apply Model" to_port="model"/>
          <connect from_port="test set" to_op="Apply Model" to_port="unlabelled data"/>
          <connect from_op="Apply Model" from_port="labelled data" to_op="Performance" to_port="labelled data"/>
          <connect from_op="Performance" from_port="performance" to_port="averagable 1"/>
          <portSpacing port="source_model" spacing="0"/>
          <portSpacing port="source_test set" spacing="0"/>
          <portSpacing port="source_through 1" spacing="0"/>
          <portSpacing port="source_through 2" spacing="0"/>
          <portSpacing port="sink_averagable 1" spacing="0"/>
          <portSpacing port="sink_averagable 2" spacing="0"/>
        </process>
        <description align="center" color="transparent" colored="false" width="126">A cross-validation evaluating a decision tree model.</description>
      </operator>
      <connect from_op="Retrieve ZNormalizedShiftedby3UnitsGangliadCacheData" from_port="output" to_op="Validation" to_port="training"/>
      <connect from_op="Validation" from_port="model" to_port="result 1"/>
      <connect from_op="Validation" from_port="averagable 1" to_port="result 2"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
      <portSpacing port="sink_result 2" spacing="0"/>
      <portSpacing port="sink_result 3" spacing="0"/>
    </process>
  </operator>
</process>
```

Listing A.2: Feature selection process using SVM. Process excludes the correlated attributes. label

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="6.5.002">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="6.5.002" expanded="true" name="Process">
    <parameter key="logverbosity" value="init"/>
    <parameter key="random_seed" value="2001"/>
    <parameter key="send_mail" value="never"/>
    <parameter key="notification_email" value=""/>
    <parameter key="process_duration_for_mail" value="30"/>
    <parameter key="encoding" value="SYSTEM"/>
    <process expanded="true">
      <operator activated="true" class="retrieve" compatibility="
          6.5.002" expanded="true" height="60" name="Retrieve
          ZNormalizedShiftedby3UnitsGangliadCacheData" width="90" x="45"
           y="30">
        <parameter key="repository_entry" value="//Local Repository/
            Path to the Imported Data Source"/>
      </operator>
      <operator activated="true" class="remove_correlated_attributes"
          compatibility="6.5.002" expanded="true" height="76" name="
          Remove Correlated Attributes" width="90" x="246" y="30">
        <parameter key="correlation" value="0.95"/>
        <parameter key="filter_relation" value="greater"/>
        <parameter key="attribute_order" value="original"/>
        <parameter key="use_absolute_correlation" value="true"/>
        <parameter key="use_local_random_seed" value="false"/>
        <parameter key="local_random_seed" value="1992"/>
      </operator>
      <operator activated="true" class="weight_by_svm" compatibility="
          6.5.002" expanded="true" height="76" name="Weight by SVM"
          width="90" x="447" y="30">
        <parameter key="normalize_weights" value="true"/>
        <parameter key="sort_weights" value="true"/>
        <parameter key="sort_direction" value="ascending"/>
        <parameter key="C" value="0.0"/>
      </operator>
      <connect from_op="Retrieve
          ZNormalizedShiftedby3UnitsGangliadCacheData" from_port="output
          " to_op="Remove Correlated Attributes" to_port="example set
          input"/>
      <connect from_op="Remove Correlated Attributes" from_port="
          example set output" to_op="Weight by SVM" to_port="example set
          "/>
      <connect from_op="Weight by SVM" from_port="weights" to_port="
          result 1"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
      <portSpacing port="sink_result 2" spacing="0"/>
    </process>
  </operator>
</process>
```

Listing A.3: SVM Classification stream for the data with three top most attributes. label

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="6.5.002">
  <context>
    <input/>
    <output/>
  <macros/>
</context>
<operator activated="true" class="process" compatibility="6.5.002"
    expanded="true" name="Process">
  <parameter key="logverbosity" value="init"/>
  <parameter key="random_seed" value="2001"/>
  <parameter key="send_mail" value="never"/>
  <parameter key="notification_email" value=""/>
  <parameter key="process_duration_for_mail" value="30"/>
  <parameter key="encoding" value="SYSTEM"/>
  <process expanded="true">
    <operator activated="true" class="retrieve" compatibility="
        6.5.002" expanded="true" height="60" name="Retrieve
        ZNormalizedShiftedby3UnitsGangliadCacheData" width="90" x="45"
         y="30">
      <parameter key="repository_entry" value="//Local Repository/
          Path to the Imported Data Source"/>
    </operator>
    <operator activated="true" class="remove_correlated_attributes"
        compatibility="6.5.002" expanded="true" height="76" name="
        Remove Correlated Attributes" width="90" x="246" y="30">
      <parameter key="correlation" value="0.95"/>
      <parameter key="filter_relation" value="greater"/>
      <parameter key="attribute_order" value="original"/>
      <parameter key="use_absolute_correlation" value="true"/>
      <parameter key="use_local_random_seed" value="false"/>
      <parameter key="local_random_seed" value="1992"/>
    </operator>
    <operator activated="true" class="weight_by_svm" compatibility="
        6.5.002" expanded="true" height="76" name="Weight by SVM"
        width="90" x="380" y="30">
      <parameter key="normalize_weights" value="true"/>
      <parameter key="sort_weights" value="true"/>
      <parameter key="sort_direction" value="ascending"/>
      <parameter key="C" value="0.0"/>
    </operator>
    <operator activated="true" class="select_by_weights"
        compatibility="6.5.002" expanded="true" height="94" name="
        Select by Weights" width="90" x="514" y="30">
      <parameter key="weight_relation" value="top k"/>
      <parameter key="weight" value="1.0"/>
      <parameter key="k" value="3"/>
      <parameter key="p" value="0.5"/>
      <parameter key="deselect_unknown" value="true"/>
      <parameter key="use_absolute_weights" value="true"/>
    </operator>
    <operator activated="true" class="x_validation" compatibility="
        5.0.000" expanded="true" height="112" name="Validation" width=
        "90" x="648" y="30">
      <parameter key="create_complete_model" value="false"/>
      <parameter key="average_performances_only" value="true"/>
      <parameter key="leave_one_out" value="false"/>
      <parameter key="number_of_validations" value="10"/>
```

```
        <parameter key="sampling_type" value="2"/>
        <parameter key="use_local_random_seed" value="false"/>
        <parameter key="local_random_seed" value="1992"/>
        <process expanded="true">
          <operator activated="true" class="support_vector_machine" compatibility="6.5.002" expanded="true" height="112" name="SVM" width="90" x="179" y="30">
            <parameter key="kernel_type" value="dot"/>
            <parameter key="kernel_gamma" value="1.0"/>
            <parameter key="kernel_sigma1" value="1.0"/>
            <parameter key="kernel_sigma2" value="0.0"/>
            <parameter key="kernel_sigma3" value="2.0"/>
            <parameter key="kernel_shift" value="1.0"/>
            <parameter key="kernel_degree" value="2.0"/>
            <parameter key="kernel_a" value="1.0"/>
            <parameter key="kernel_b" value="0.0"/>
            <parameter key="kernel_cache" value="200"/>
            <parameter key="C" value="0.0"/>
            <parameter key="convergence_epsilon" value="0.001"/>
            <parameter key="max_iterations" value="100000"/>
            <parameter key="scale" value="true"/>
            <parameter key="calculate_weights" value="true"/>
            <parameter key="return_optimization_performance" value="true"/>
            <parameter key="L_pos" value="1.0"/>
            <parameter key="L_neg" value="1.0"/>
            <parameter key="epsilon" value="0.0"/>
            <parameter key="epsilon_plus" value="0.0"/>
            <parameter key="epsilon_minus" value="0.0"/>
            <parameter key="balance_cost" value="false"/>
            <parameter key="quadratic_loss_pos" value="false"/>
            <parameter key="quadratic_loss_neg" value="false"/>
            <parameter key="estimate_performance" value="false"/>
          </operator>
          <connect from_port="training" to_op="SVM" to_port="training set"/>
          <connect from_op="SVM" from_port="model" to_port="model"/>
          <connect from_op="SVM" from_port="weights" to_port="through 1"/>
          <portSpacing port="source_training" spacing="0"/>
          <portSpacing port="sink_model" spacing="0"/>
          <portSpacing port="sink_through 1" spacing="0"/>
          <portSpacing port="sink_through 2" spacing="0"/>
        </process>
        <process expanded="true">
          <operator activated="true" class="apply_model" compatibility="5.0.000" expanded="true" height="76" name="Apply Model" width="90" x="45" y="30">
            <list key="application_parameters"/>
            <parameter key="create_view" value="false"/>
          </operator>
          <operator activated="true" class="performance" compatibility="5.0.000" expanded="true" height="76" name="Performance"
```

```xml
            width="90" x="179" y="30">
            <parameter key="use_example_weights" value="true"/>
          </operator>
          <connect from_port="model" to_op="Apply Model" to_port="model"/>
          <connect from_port="test set" to_op="Apply Model" to_port="unlabelled data"/>
          <connect from_op="Apply Model" from_port="labelled data" to_op="Performance" to_port="labelled data"/>
          <connect from_op="Performance" from_port="performance" to_port="averagable 1"/>
          <portSpacing port="source_model" spacing="0"/>
          <portSpacing port="source_test set" spacing="0"/>
          <portSpacing port="source_through 1" spacing="0"/>
          <portSpacing port="source_through 2" spacing="0"/>
          <portSpacing port="sink_averagable 1" spacing="0"/>
          <portSpacing port="sink_averagable 2" spacing="0"/>
        </process>
        <description align="center" color="transparent" colored="false" width="126">A cross-validation evaluating a decision tree model.</description>
      </operator>
      <connect from_op="Retrieve ZNormalizedShiftedby3UnitsGangliadCacheData" from_port="output" to_op="Remove Correlated Attributes" to_port="example set input"/>
      <connect from_op="Remove Correlated Attributes" from_port="example set output" to_op="Weight by SVM" to_port="example set"/>
      <connect from_op="Weight by SVM" from_port="weights" to_op="Select by Weights" to_port="weights"/>
      <connect from_op="Weight by SVM" from_port="example set" to_op="Select by Weights" to_port="example set input"/>
      <connect from_op="Select by Weights" from_port="example set output" to_op="Validation" to_port="training"/>
      <connect from_op="Validation" from_port="model" to_port="result 1"/>
      <connect from_op="Validation" from_port="averagable 1" to_port="result 2"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
      <portSpacing port="sink_result 2" spacing="0"/>
      <portSpacing port="sink_result 3" spacing="0"/>
    </process>
  </operator>
</process>
```

B

Efficiency of SVM classifier for different sets of input data

Input Data		True 1	True 0	Class Precision
All, 18 Attributes for input	Prediction 1	2593	16	99.39%
	Prediction 0	30	0	0.00%
	class recall	98.86%	0.00%	Accuracy 98.26%
CPU System, Load Fifteen TCP Established Top 3 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
CPU System and Load Fifteen With Gradient 4 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
CPU System and TCP Established With Gradient 4 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
Load Fifteen and TCP Established With Gradient 4 Attributes for Input	Prediction 1	2622	16	99.39%
	Prediction 0	1	0	0.00%
	class recall	99.96%	0.00%	Accuracy 99.36%
CPU System With Gradient 2 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
Load Fifteen With Gradient 2 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
TCP Established With Gradient 2 Attributes for Input	Prediction 1	2623	16	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%

C

Efficiency of ANFIS classifier for different sets of input data

Input Data		True 1	True 0	Class Precision
CPU System, Load Fifteen TCP Established Top 3 Attributes for Input	Prediction 1	1310	3	99.77%
	Prediction 0	2	5	71.43%
	class recall	99.84%	62.5%	Accuracy 99.62%
CPU System and Load Fifteen With Gradient 4 Attributes for Input	Prediction 1	1307	4	99.69%
	Prediction 0	5	4	44.44%
	class recall	99.62%	50.00%	Accuracy 99.32%
CPU System and TCP Established With Gradient 4 Attributes for Input	Prediction 1	1300	3	99.77%
	Prediction 0	12	5	29.41%
	class recall	99.09%	62.5%	Accuracy 98.86%
Load Fifteen and TCP Established With Gradient 4 Attributes for Input	Prediction 1	1308	4	99.7%
	Prediction 0	4	4	50.00%
	class recall	99.7%	50.00%	Accuracy 99.39%
CPU System With Gradient 2 Attributes for Input	Prediction 1	1308	5	99.62%
	Prediction 0	4	3	42.86%
	class recall	99.69%	37.5%	Accuracy 99.32%
Load Fifteen With Gradient 2 Attributes for Input	Prediction 1	1312	8	99.39%
	Prediction 0	0	0	0.00%
	class recall	100.00%	0.00%	Accuracy 99.39%
TCP Established With Gradient 2 Attributes for Input	Prediction 1	1308	4	99.7%
	Prediction 0	4	4	0.00%
	class recall	99.7%	50.00%	Accuracy 99.39%

D

ANFIS parameters and structure after training

```
[System]
Name='ANFIS'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='prod'
OrMethod='probor'
ImpMethod='prod'
AggMethod='sum'
DefuzzMethod='wtaver'

[Input1]
Name='input1'
Range=[1.138838 7.027855]
NumMFs=3
MF1='in1mf1':'gauss2mf',[0.500167052036675 0.25548545
0.488808731716495 2.01677511182972]
MF2='in1mf2':'gauss2mf',[0.497667491914805 3.20042629930766
0.500179055599698 4.96670378894294]
MF3='in1mf3':'gauss2mf',[0.500346981253454 6.14443764905214
0.500167052036675 7.91120755]

[Input2]
Name='input2'
```

```
Range=[1.454491 5.520699]
NumMFs=3
MF1='in2mf1':'gauss2mf',[0.345351909890555 0.8445598
0.343860868407268 2.06456838380518]
MF2='in2mf2':'gauss2mf',[0.343011986313515 2.8794552544224
0.34394228525347 4.0967301465124]
MF3='in2mf3':'gauss2mf',[0.394121448390163 4.89489214250976
0.345351909890555 6.1306302]

[Input3]
Name='input3'
Range=[-0.880052 6.575568]
NumMFs=3
MF1='in3mf1':'gauss2mf',[0.633222060066336 -1.998395
0.657146029020811 0.252312731554805]
MF2='in3mf2':'gauss2mf',[0.631438844619175 1.73191311952562
0.632597470475098 3.96580425584288]
MF3='in3mf3':'gauss2mf',[0.633689815260645 5.45711062868705
0.633222060066336 7.693911]

[Output1]
Name='output'
Range=[0 1]
NumMFs=27
MF1='out1mf1':'constant',[-8.07606484251085]
MF2='out1mf2':'constant',[1.0214719070848]
MF3='out1mf3':'constant',[-0.96815561790751]
MF4='out1mf4':'constant',[-0.436431820825542]
MF5='out1mf5':'constant',[0.972881397855334]
MF6='out1mf6':'constant',[1.13075402662828]
MF7='out1mf7':'constant',[-324.480831090797]
MF8='out1mf8':'constant',[1.19425824054878]
MF9='out1mf9':'constant',[-4.6031235314334]
MF10='out1mf10':'constant',[1.27405858376483]
MF11='out1mf11':'constant',[1.00246580816994]
MF12='out1mf12':'constant',[0.958088988213647]
MF13='out1mf13':'constant',[1.1851566213193]
MF14='out1mf14':'constant',[1.00194407144311]
MF15='out1mf15':'constant',[0.998210284800305]
MF16='out1mf16':'constant',[1.01642207961165]
MF17='out1mf17':'constant',[0.987885594676387]
MF18='out1mf18':'constant',[1.00917226184839]
MF19='out1mf19':'constant',[0.00616230973994245]
```

```
MF20='out1mf20':'constant',[0.706906959025678]
MF21='out1mf21':'constant',[0.00109486488388064]
MF22='out1mf22':'constant',[2.52190271722454]
MF23='out1mf23':'constant',[1.00010469139032]
MF24='out1mf24':'constant',[1.31823381648714]
MF25='out1mf25':'constant',[-8.79217318197809]
MF26='out1mf26':'constant',[1.01449980885013]
MF27='out1mf27':'constant',[0.0345207843342135]

  [Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1
```