www.qualiMaster.eu

QualiMaster

A configurable real-time Data Processing Infrastructure mastering autonomous Quality Adaptation

Grant Agreement No. 619525

Deliverable D5.1

Work-package	WP5: The QualiMaster Adaptive Real-Time Stream Processing Infrastructure
Deliverable	D5.1: QualiMaster Infrastructure Set-up
Deliverable Leader	TSI
Quality Assessors	Stefan Burkard, Oliver Pell
Estimation of PM spent	5 PM
Dissemination level	PU
Delivery date in Annex I	July 31, 2014
Actual delivery date	August 1, 2014
Revisions	4
Status	Final
Keywords:	Infrastructure, Architecture

Disclaimer

List of Authors

Partner Acronym	Authors
LUH	Mohammad Alrifai
	Claudia Niederée
SUH	Holger Eichelberger
	Cui Qin
TSI	Ekaterini Ioannou
	Evripidis Sotiriadis

Table of Contents

Executive summary

This deliverable reports the set-up and current status of the QualiMaster infrastructure. We present the infrastructure in terms of its set-up, its architecture and interactions between the components of the architecture. These primarily include the processing elements, pipeline execution, reconfigurable hardware, and monitoring and adaptation mechanisms. We also provide an overview of the external systems incorporated in the QualiMaster infrastructure and discuss the advantages/functionalities that we expect these systems will provide. Furthermore, we report on the execution environment that is currently available to all the project partners and discuss the current status of the infrastructure as well as the related scheduled future plans. This report is just part of the deliverable. The QualiMaster infrastructure core as it is described in this deliverable is running as part of the IT infrastructure of TSI and is available for the further developments in the project.

D5.1 is closely related to deliverable D1.1, since D1.1 provided the (initial) requirements that drove the set-up of the infrastructure. Also note that D5.1 is the first deliverable of WP5 and it will be followed by deliverable D5.2 (Basic QualiMaster Infrastructure), D5.3 (QualiMaster Infrastructure V1), and D5.4 (QualiMaster Infrastructure V2). These future WP5 deliverables will document our efforts on improving and extending the QualiMaster infrastructure.

1 Introduction

This deliverable describes and discusses the QualiMaster infrastructure. This includes the set-up of the infrastructure as well as the interactions between the incorporated layers, tools, and external systems, such as a real-time stream processing framework or reconfigurable computing hardware. Our presentation of the infrastructure also provides an overview of the external systems we utilize and their advantages and functionalities. Another important aspect discussed in this document is the current status of the realization of the infrastructure and future plans.

This report is just part of the deliverable. The QualiMaster infrastructure core as it is described in this deliverable is running as part of the IT infrastructure of TSI and is available for the further developments in the project

The description of the QualiMaster infrastructure, as provided in this deliverable, mainly depicts the current status that will evolve during the project. Thus, it could be that some of the layers, tool and external systems will change due to the constant monitoring of alternatives, e.g., decisions for execution systems. These extensions and modifications will be reported in deliverables D5.2, D5.3, and D5.4.

In the following paragraphs we discuss the requirements considered for creating the presented QualiMaster infrastructure (Section 1.1) followed by a description of the QualiMaster Priority Pipeline (Section 1.2), which will be the development priority during the next months.

1.1 Requirements driving the infrastructure design

Guiding our efforts for creating the design and set-up of the QualiMaster infrastructure and its architecture we have considered and analyzed the requirements collected in WP1. More specifically, we used deliverable D1.1 that presents a collection of requirements for the QualiMaster project and the QualiMaster Applications for systemic risk analysis in the financial domain. The D1.1 deliverable also presents the actors who will interact with the applications, an initial description of individual use cases that encode the interactions among components, requirements for the data streams to be processed and the algorithms to be applied in a data analysis pipeline for systemic risk calculation.

One of the main requirements defined in deliverable D1.1, which we have considered during the set-up of the QualiMaster infrastructure, is the **real-time data stream processing**. QualiMaster will receive real-time data, e.g., financial data that is typically composed of numbers and Web data that is composed of more complex data types. This data is expected to go through real-time analysis, such as systemic risk analysis, with the results given to the QualiMaster applications.

As explained in the D1.1 deliverable, QualiMaster must be able to perform large-scale stream processing and accelerate a wide range of data intensive algorithms. To assist this goal, the consortium will exploit **hardware-based processing**. However, this requires considering hardware-based implementations of (parts of) the financial processing algorithms, being able to combine software- and hardware-based processing, and using the kernels (i.e., reconfigurable hardware) in an optimal manner.

A requirement related to the real-time data stream processing is the **necessity to adapt the processing** to the needs and the runtime demands of the actual data streams. This basically implies the need to incorporate autonomous activities for maintaining the actual quality of the data analysis and the efficiency of the use of the physical computing resources. This quality-aware adaptation needs an architecture that supports the monitoring of the execution state, quickly deciding and enacting adaptation decisions.

The analysis algorithms for systemic risk require historical data of the specific domain. This means that QualiMaster needs to be able to **batch process the historical data**. Integrating historical data with specific data analysis processes requires not only being able to maintain and access the data, but also having the capability to incorporate the analysis of static data sources during the real-time stream processing.

It is intended that the QualiMaster infrastructure will integrate various systems that need to work together and adapt to the current processing needs (as explained above). In order to save time, effort, computational resources, and to support the uptake of the platform in other setting or domains we need **flexibility through configuration**, i.e., being able to easily setup the platform itself as well as the data analysis to be executed on the platform.

Relation to Existing Projects. Part of the requirements introduced in deliverable D1.1 can be satisfied by existing projects. The consortium has detected two such projects, namely TrendMiner[1] and Juniper[2].

TrendMiner focuses on providing real-time methods for cross-lingual mining and summarization of large-scale stream media. The project's case studies include financial decision support, which resembles the high level goal of QualiMaster.

[1] http://www.trendminer-project.eu/

[2] http://www.juniper-project.org/

However, the focus of QualiMaster goes beyond this, and more specifically on enabling autonomous proactive, reflective, and cross-pipeline adaptation, exploitation of families of approximate algorithms with different quality/performance tradeoffs, and achieving scalability using reconfigurable hardware. Juniper focuses on providing efficient and real-time exploitation of large streaming data sources and stored data. The project uses financial and web streaming case studies, which seems similar to the domains used in QualiMaster. The QualiMaster consortium is currently building up a connection with the main actors from the Juniper project, in order to learn more about their project results and subsequently how the results and insights from the Juniper project can be used in the QualiMaster project.

We are interested to exploit some results from these projects, such as the batch processing tools for Twitter text analysis that will be developed in TrendMiner. For this reason, the consortium will monitor the progress of these projects and investigate following up on interesting aspects.

1.2 The QualiMaster Priority Pipeline

To prioritize our efforts with respect to the infrastructure development and implementation within the first year of the project, the consortium decided to define and focus on a *QualiMaster Priority Pipeline*. The goal is to have all the functionalities included in this priority pipeline running on the QualiMaster infrastructure by the end of the first year, i.e., December 2014. It is expected that this integrative way of prioritizing the work based on a joint priority pipeline will foster integration and will help in discovering important integration challenges early in the project.

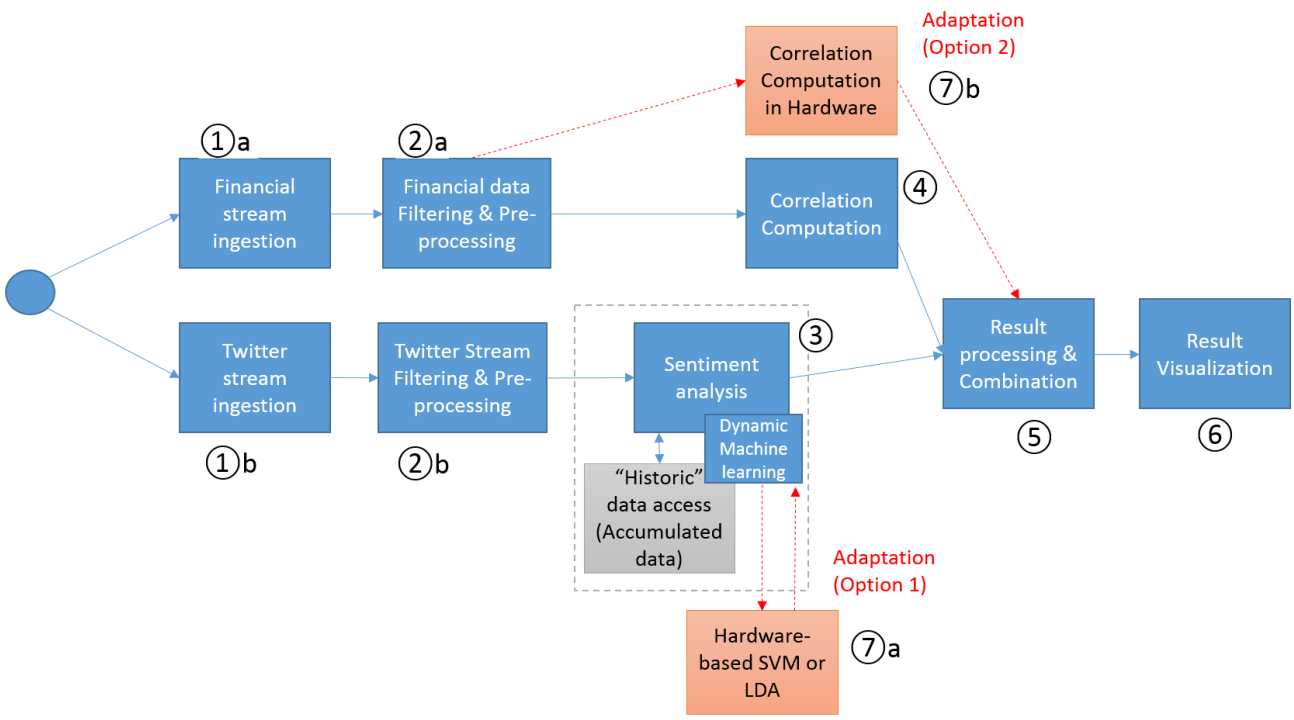

Figure 1: Illustration of the QualiMaster priority pipeline.

Creating a priority pipeline ensures that there are early demonstrators, which go beyond showing individual component functionality. This is expected to be helpful for discussing the QualiMaster approach with other experts as well as for presenting the novel capabilities of QualiMaster to other interested parties, such as the customers of the SMEs involved in the QualiMaster project, or regulatory bodies in the financial domain.

Having those goals in mind, the selection and design of the priority pipeline considerate the following desirable properties:

- The pipeline must emphasize the most challenging integration tasks: it should be adequate for showing the interconnection between hardware- and software-based stream processing.
- Processing should exploit and combine data streams from the financial domain and from the Social Web.
- It should be driven by the needs of the targeted QualiMaster application scenarios.
- It must illustrate the combination of processing stored data with real-time streams.
- It should show a first example of dynamic pipeline adaptation.

Incorporating these properties, we have defined a first version of the priority pipeline (shown in Figure 1). The preliminary QualiMaster Priority pipeline will consist of the following steps:

1. **Data stream ingestion:**
    a. Ingestion of the Twitter stream
    b. Ingestion of the financial data stream
2. **Data Stream Preprocessing and Filtering:** There are two such steps, one for financial data stream (Step 2a) and another for the social media stream (Step 2b).
3. **Social Media Stream:** Here, it is planned to do real time sentiment analysis (e.g., for key market players) and also include the accumulated sentiment information (historic data). The idea is to collect sentiment trends, which might play a role in assessing and predicting financial developments.
4. **Correlation Computation:** In this step, correlations between multiple market players are computed. This step will make heavy use of the high performance processing aspect of the QualiMaster infrastructure.
5. **Result Processing & Combination:** This step contains post-processing of the results computed from the two streams and the activities for result combination.
6. **Result Visualization:** In this step, the results of the analysis of the financial data stream and the social Web stream will be visualized in a way that this will support the decision making of the financial stakeholders.
7. **Pipeline adaptation:** For dynamic adaptation, it is planned to “switch” one part of the computation from software-based processing to hardware-based processing. This will be based on measuring the pipeline performances and making an on-the-fly adaptation decision. Two possible switching mechanisms are currently under discussion: a) exploiting a hardware-based SVM (Support Vector Machines) or LDA (Latent Dirichlet Allocation) in Twitter analysis, and b) performing part of the correlation computation in reconfigurable hardware.

Note that the described priority pipeline is an implementation and integration plan targeting the end of the first year of the project. The first individual parts for the pipeline are currently under development. Considerable implementation and integration work is still required, but it is expected that the planned priority pipeline (maybe still with some minor modifications) will be completed by the end of the year.

As a preparation step for the QualiMaster priority pipeline, two bootstrapping pipelines (comparable to “hello world” programs) have been defined to assist in the testing of the created and configured infrastructure. These basic pipelines are

described in Section 5.1 and are part of the infrastructure setup that has been completed for D5.1.

1.3 Structure of the deliverable

The remaining sections of this deliverable are as follows: Section 2 provides an overview to the QualiMaster infrastructure, including a discussion of the data flow among the infrastructure tools and layers. Section 3 presents the external systems that we have incorporated in QualiMaster and explains the benefits and functionalities we expect to have given these systems. Section 4 provides the details of the infrastructure layers, including the configuration, startup time, and runtime lifecycle phases. Section 5 provides a discussion about the current status of the infrastructure along with the created execution environment, and finally Section 6 provides conclusions for this deliverable and gives an overview of the future plans.

2 The QualiMaster Infrastructure

In this section we present and discuss the infrastructure for QualiMaster. We begin with an overview of the infrastructure (Section 2.1) and then discuss the flow between the infrastructure's layers (Section 2.2).

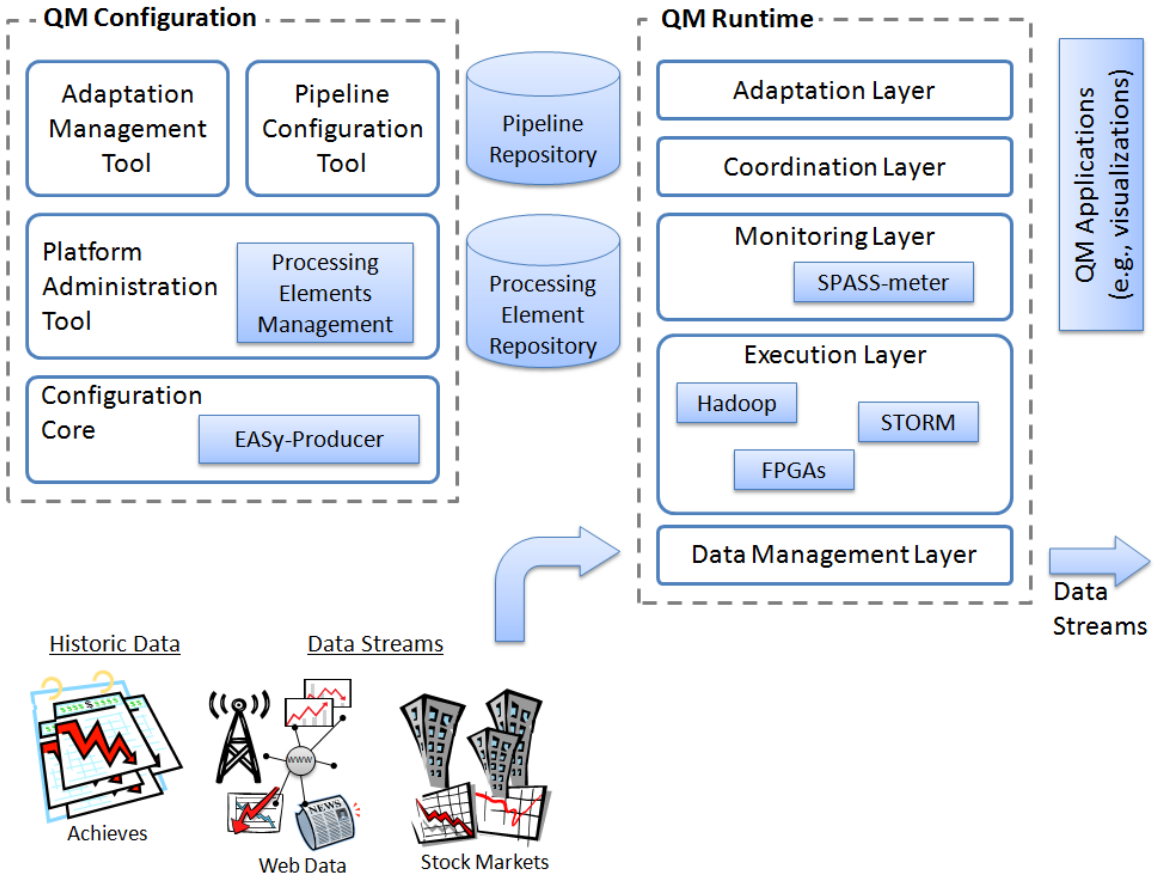

Figure 2: Illustration of the QualiMaster Infrastructure.

2.1 Overview

Figure 2 shows an illustration of the QualiMaster Infrastructure. The QualiMaster input can be: (i) real-time data streams, and (ii) historical data that the infrastructure either maintains in local stores or receives by connecting to remote stores. In the QualiMaster application cases, real-time data will come from stock markets or Web data, such as Twitter, while historical data will encompass stock market data and social media data as required by the individual analysis pipelines. In particular, SPRING will provide a remote data store on historical financial data, which cannot be directly managed by the infrastructure due to license restrictions that apply for the financial data. The goal of the infrastructure is to perform real-time processing

of the received data, for example to perform systemic risk analysis of the received stock markets.

The QualiMaster Infrastructure has three ***lifecycle phases***.

1. The **configuration time** that utilizes the infrastructure tooling specifying the configuration for a certain application setting (e.g., financial data processing) and for the pipelines to be executed. The infrastructure tooling consists of the Pipeline Configuration tool (see D1.1, Section 5.1), the Adaptation Manager tool (see D1.1, Section 5.2), and the Platform Administration tool (see D1.1, Section 5.3). The ultimate action at configuration time is the derivation of an instantiated version of the QualiMaster platform and the pipelines to be executed in order to maximize the runtime performance.
2. The **startup** of the QualiMaster platform. This lifecycle phase includes initial (re)configuration of hardware, e.g., upload of the laid out hardware algorithms, the deployment and startup of programs controlling the communication with the reconfigurable hardware (the so called “host” in Maxeler infrastructures), and the startup of the (configured) real-time data processing pipeline and the historical data processing.
3. The **runtime** lifecycle phase that involves the actual processing of the streaming data. We have two processing options: (i) *autonomous* that performs the enactment of adaptation actions determined by the reactive, proactive, or (automatic) reflective adaptation, and (ii) *manual* that performs the enactment of manual decisions or actions suggested by the reflective adaptation but with manual approval.

Please note that in Section 4 we provide a detailed presentation and discussion of the layers and tools composing each of these three lifecycle phases.

2.2 Flow between layers and tools

Consider now that the QualiMaster infrastructure is already running, i.e., we are in the runtime lifecycle phase. A QualiMaster user has specified his/her desired financial analysis processes along with the stocks he would like to monitor, for example the ones described by the user scenarios presented in Section 3.2 “Application Use Cases for Systemic Risk Assessment for Institutional Financial Clients” of deliverable D1.1.

To perform the processing encoded in such a financial analysis, the QualiMaster infrastructure involves a number of layers. These layers and their responsibilities are:

- **Data Management Layer -** manages historical data relevant to the actual running pipelines. Here, we distinguish between the management of *raw data*

(input tuples to a pipeline) and the management of *processed data (including final results)*. Raw data may be used as input to several pipelines running on the QualiMaster platform. Thus, storing raw data is a common task in order to avoid repeated storage of the same information. However, storing all raw input data may not be feasible in all application cases. Thus, the QualiMaster infrastructure will provide support to tailor the input aspect of the Data Management Layer in terms of configuration options, e.g., to disable storage for a certain data source or to select the storage strategy. Regarding processed data, the Pipeline Designer will be able to specify as part of the pipeline design, how and what data shall actually be stored, e.g., in terms of generic built-in processing elements, which pass the related data tuples to the Data Management Layer for storing.

- **Execution Layer** - includes the *execution systems* that perform the actual execution of the data analysis algorithms, either in terms of a real-time stream processing topologies on Apache Storm, the execution of hardware-based algorithms on specialized hardware, such as MAX dataflow engines, or the execution of data analysis algorithms on historical mass data in a batch manner using Apache Hadoop.
- **Monitoring Layer** - (distributed) surveillance of the execution layer in order to obtain readings of quality parameters for the actual execution of pipelines. This layer unifies and - if required - aggregates monitored information from different sources, such as generated or instrumented code, and specific modules realized for the individual execution systems of the execution layer.
- **Coordination Layer** - enacting adjustments to the data analysis pipelines at runtime, such as selecting a specific data processing algorithm from an algorithm family. One adjustment may imply further enactments, e.g., when switching the execution from software to hardware, two execution systems are involved. Enacting changes efficiently across multiple execution systems is handled by the coordination layer. Furthermore, the coordination layer may use changes in monitoring through the Monitoring Layer at runtime, e.g., to change monitoring priorities.
- **Adaptation Layer** - adaptive decision making based on monitoring the actual execution of pipelines (as provided by the Monitoring layer). Decisions made by this layer are enacted by the Coordination Layer.

In the following paragraphs, we elaborate on the most important interactions among the individual layers.

The pipeline encoding the desired financial analysis is initially given to the Coordination Layer. The Coordination Layer processes and deploys the pipeline accordingly (e.g., a JAR file to Apache Storm or a .maxj file to the dataflow engines)

and sends the appropriate commands to control the execution in the Execution Layer, e.g., starting the pipeline. The Execution Layer is responsible for executing these commands using distributed real-time computation system, such as Apache Storm, on the processing nodes it incorporates, such as computer cluster nodes and High Performance Computing (HPC) reconfigurable nodes. The reconfigurable nodes are server-class HPC systems with reconfigurable dataflow compute engines (DFEs), which can be integrated in a cluster infrastructure. Note that the execution also involves receiving the streaming data and handling the final results, which is performed through the Data Management Layer.

The Monitoring Layer is constantly observing the processing performed by the Execution Layer as well as the performance of the systems it uses (e.g., reconfigurable hardware) and collects statistics that are maintained locally in the Monitoring Layer. The collected information is given to the Adaptation Layer for deciding reactive, proactive, or (automatic) reflective adaptation. More specifically, the Adaptation Layer will analyze the information received by the Monitoring Layer and (if needed) requests enacting the commands on the pipelines encoding the specific financial analysis via the Coordination Layer. The modified commands are passed to the Execution Layer for processing on the actual execution systems.

3 External Systems incorporated in QualiMaster

We now provide a discussion of the required functionalities that are provided by external systems and will be utilized by the QualiMaster infrastructure. For each functionality we provide a discussion of the available systems and explain the reasons for which we selected to incorporate the specific external system.

3.1 Distributed real-time Computation System

QualiMaster needs to enable real-time computation over a large number of data streams. Such functionality is currently available by a few existing streaming frameworks. The most popular frameworks for real-time computation are StreamMine3G[3], Yahoo! S4[4], StreamMapReduce [BMK+11], and Apache Storm[5]. The QualiMaster infrastructure will incorporate Apache Storm – the following paragraphs discuss the reasons for this selection.

The S4 system by Yahoo, which is short for “Simple Scalable Streaming System” [NRNK10], is a stream-based cloud computing platform. It offers a simple dataflow-based programming interface for deploying concurrent algorithms over large clusters of nodes. Upon deployment of a streaming algorithm, the processing elements included in the graph representing the algorithm are instantiated at various nodes. S4 routes the events that are to be processed to these nodes.

Both Storm and S4 are released under Apache license, and are both plain Java-based, i.e., platform-independent. They are both at incubation phase, although quite mature. We have selected Storm over S4 for two main reasons:

a) Storm has a substantially more vibrant community (both developers and users) than S4. For example, the developers list of Storm had 3564 messages in the first semester of 2014, whereas the corresponding S4 mailing list had only 37 messages in the corresponding period. (A similar ratio was observed comparing the users mailing lists of the two computation systems.) We consider this to be important for QualiMaster for two reasons. First, it is much more probable to get support in a project with a vibrant community and with a large user base, in case something goes wrong, e.g., there is a bug in either Storm or S4. Second, any possible contributions made by QualiMaster will be more influential, as they will be applicable to a much larger user base.
b) Even though the S4 project had substantial momentum in 2011 and 2012, the last version of S4 is released more than one year ago (June 2013). Storm

[3] https://streammine3g.inf.tu-dresden.de/trac
[4] http://incubator.apache.org/s4/
[5] https://storm.incubator.apache.org/

on the other hand has regular updates, with its last version released less than a month ago (June 2014). This is attributed to the much larger user base of Storm.

Two other possible systems for real-time computation are StreamMine3G [MFB11] and StreamMapReduce [BMK+11]. StreamMine3G focuses on efficient support for fault tolerance through active replication. All the Event Stream Processing (ESP) operators are replicated, so that any failure of an operator is masked by a replica. StreamMapReduce relies on an extension of the MapReduce programming model to allow processing of unbounded event streams and it enables higher parallelization by allowing an overlap between the map and reduce phases.

StreamMine3G and StreamMapReduce are closed-source and constructed by a few individuals, without a backing community. In addition, deploying onto different platforms is easier with Storm since this is written in plain Java, which is platform-independent.

3.2 Distributed Storage and Batch Processing

According to the user requirements (D1.1), the QualiMaster infrastructure (in addition to real-time streams) must support the integration of historical data sources and data processing. This also includes queries over historical data (REQ-DS3).

Therefore, QualiMaster will provide the possibility to store the raw data coming from live streams of both financial markets and the social web. In addition, the infrastructure will enable the storage of the processed data (i.e., results from real-time processing) in the QualiMaster storage system for future analysis and visualization tasks. As part of the configuration, the pipelines designer will specify which data should be stored and in which format (possibly as part of a data processing element that defines the type and structure of tuples that should be sent to the Data Management Layer for storage).

YARN[6] is one of the widely used solutions for efficient and distributed storage batch processing of large datasets across clusters of computers. It scales up from single servers to thousands of machines, each offering local computation and storage. Distributed storage is realized by its distributed file system called HDFS and distributed and parallel processing is realized by its implementation of the MapReduce framework[7]. Hadoop also detects and handles failures at the

[6] http://hadoop.apache.org/
[7] http://research.google.com/archive/mapreduce.html

application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

There are several Apache projects that provide the Hadoop project with additional support for more efficient and scalable storage and processing of big data. These include:

- HBase[8]: a distributed, scalable, big data store. It is considered as the Hadoop NoSQL database implementing Google's Bigtable model[9].
- Pig[10]: a high-level data-flow language and execution framework for parallel computation.
- Mahout[11]: a Scalable machine learning and data mining library.

Due to its maturity and wide acceptance in the industry as well as its rich portfolio of accompanying tools, the QualiMaster infrastructure will deploy Hadoop YARN as the main framework for distributed storage and batch processing of web and financial datasets. However, during the course of the project, the deployment of other emerging alternatives will be investigated.

One promising alternative for Hadoop's MapReduce implementation is the new emerging Apache Spark project[12], which provides a low-latency, memory-based computation engine for big data. It provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation. Spark can be used as a standalone solution or as part of the Hadoop system.

3.3 Reconfigurable Hardware

Field Programmable Gate Arrays (FPGAs) are programmable devices used for several purposes, such as network systems and high performance computing. FPGAs were introduced in the early 80s and were targeted for the execution of several kinds of computations. Some of these computations are image processing [HLA98, M98, LLR+99], automated target recognition [RH97], data encryption [EP00, P00, LCTL00], factoring large numbers [KM00], cryptography applications [DPR00, LMWL00], video processing [KBD03, PVH99], string pattern matching [WL99], discrete mathematics problems [SDA00, DSE98], FFT implementations [SAA95], data compression [HSM00], speech recognition [SDD00], and arithmetic applications [PR97, LC97].

[8] http://hbase.apache.org/
[9] http://research.google.com/archive/bigtable.html
[10] http://pig.apache.org/
[11] http://mahout.apache.org/
[12] http://spark.apache.org/

FPGAs consist of several structural elements, such as lookup tables, block Random Access Memories (RAMs), digital signal processing units, and many buses for interconnection. These elements can be combined in several different ways to implement complex structures in hardware, therefore allowing the execution of any kind of computation. This flexibility allows high parallelism in data processing, which can offer performance with significant speedup against conventional computers, even if the FPGAs clock rate is about one tenth of a conventional processor.

Computations with FPGAs for High Performance Computing (HPC)

High performance computing platforms focus on accelerating applications. One of the best solutions to the acceleration problem is the use of hardware accelerators that augment processors with application specific coprocessors. Reconfigurable computing is the basis of many modern coprocessor designs.

Reconfigurable computing is a modern computer architecture trend that combines the software with high performance reconfigurable hardware fabrics. The main benefits of heterogeneous reconfigurable computing systems are given by offloading the computationally intensive part of the algorithm onto a hardware-based platform.

They offer many major capabilities in terms of High Performance Computing (HPC): accelerating application specific processes, scalability, high-bandwidth, and low power.

Reconfigurable platforms can implement any number of different architectures. One of the most promising for achieving high performance is mapping applications to highly parallel dataflow architectures, such as those implemented by Maxeler Dataflow Engines (DFEs). These utilize an explicitly parallel programming model, making it easy to access both deep pipeline parallelism and data vector parallelism, providing high performance. Reconfigurable platforms also offer high adaptability, since the chips can be reconfigured to implement different algorithms on demand. One restriction of all reconfigurable architectures lies in the number of resources. However, this problem seems to be solved in modern FPGA-based platforms, such as Maxeler dataflow computers, with the use of multiple reconfigurable dataflow engines that work in parallel.

Next, the data transmission rate is a key factor for the HPC systems. High-end reconfigurable platforms, such as Maxeler platforms, offer a combination of

reconfigurable logic and high-bandwidth, as well as low-latency interfaces to both the main processor and the system's memory. Maxeler systems can support many GB/s of bandwidth from the CPU to the DFEs, between DFEs (via a proprietary MaxRing interconnect) and between DFEs and the network (via dedicated 40Gbit Ethernet links).

Last, another important factor of reconfigurable computing systems is the low power consumption per unit of performance. Although computer clusters comprise low cost general purpose CPUs, large clusters require large data centers that are power hungry and expensive to operate over time. On the other hand, reconfigurable platforms have proved that they can offer orders-of-magnitude better energy consumption per computation versus conventional computing.

Concluding, reconfigurable computing is an important tool for high performance computing, because it achieves impressive performance advantages with low power consumption. On the other hand, the resources of the reconfigurable platforms can be a restriction when a certain level of scalability and parallelization is needed by an application. Thus, high-end platforms coupled with multiple reconfigurable devices and high data transmission links have been selected.

In summary, reconfigurable computing systems provide a powerful approach to quickly perform highly complex calculations, in particular over real-time data streams. The financial domain, i.e. the application and demonstration domain in the QualiMaster project, embraced high-performance reconfigurable computing, e.g., for the realization of complex and high-frequency processing, which is closely integrated into major stock exchanges. The QualiMaster project aims at identifying complex bottleneck streaming calculations for the software-based systems on the financial and the machine learning domain. These calculations will be mapped on reconfigurable computing platforms, which will be part of a unified cluster infrastructure of computer nodes, exploiting the capabilities of reconfigurable computing for real-time adaptive data stream processing.

3.4 Flexibility by Configuration

In QualiMaster, flexibility aims at supporting a stakeholder in easily defining the analysis tasks to be carried out by the QualiMaster platform, in particular to define their own analysis tasks using their own algorithms. Furthermore, at a larger scale, such a flexible configuration enables and supports a stakeholder to apply the QualiMaster solutions even to other application domains. However, optimizing the QualiMaster platform for the execution is a complex task which requires detailed knowledge about different interconnected elements ranging from the execution systems, the monitoring up to the adaptation layer.

Software Product Line Engineering (SPLE) [LSR07, PBL05, CN02] is a successful approach to mass customization, configuration, tailoring, and reuse of software. SPLE is widely and successfully applied in different industrial settings, such as embedded systems or information systems and helps in managing complexity of configurations, reducing time-to-market and reducing development efforts. Basically, a Software Product Line (SPL) does not aim at developing a single software system, rather it develops a family of similar, but differently configured systems. Coping with the complexity of such configurations, e.g., several thousands of decisions with interdependencies are reported in literature [BRN+13], is one of the key topics in SPLE. Deriving a specific member from a family includes the specification of its configuration, the validation of the configuration, and, finally, turning the configuration into artifacts, e.g., modifying, deleting or generating configuration files or source code as well as compiling and packaging the derived product.

In order to be applied successfully, SPLE requires specific activities in all phases of the software lifecycle, such as architectural patterns, implementation techniques, or adequate testing strategies. In addition, SPLE specific activities are performed, such as product line scoping (determining and focusing the reuse on the most beneficial aspects), or variability management (modeling and managing the configuration opportunities and their interdependencies). In summary, SPLE allows to efficiently and flexibly configure, tailor and reuse software systems using adequate methods, models and tooling. While traditional SPLE focuses on pre-runtime customization, Dynamic Software Product Lines (DSPL) [HHPS08] aim at runtime (re-)configuration and is one approach to implement adaptive systems.

In QualiMaster, traditional SPLE methods and techniques are applied to flexibly tailor the QualiMaster platform to different domains or application areas. This implies a specific variability model, i.e., a configuration meta model, of the configuration opportunities in QualiMaster ranging from the hardware to execute the pipelines up to the specification of the adaptivity and the pipelines to be executed. Based on a specific configuration meta model, the configuration of a particular application environment for the QualiMaster platform can be described and the platform as well as the pipelines to run on that platform can be instantiated. To ease the configuration task for infrastructure users, this instantiation step is done automatically and, further, enables to optimize the platform, such as avoiding unnecessary monitoring. However, describing runtime mechanisms, such as the adaptation in terms of a configuration, requires that runtime information can be used in the specification before the actual runtime of the platform. In other words, specific configuration options are left open when determining the configuration to

startup the QualiMaster platform, but filled with actual values and validated at runtime.

EASy-Producer[13] is a SPLE tool, which provides support for the development of SPL, and, in particular, facilities the most recent trends and concepts in SPLE, such as large-scale product lines, (composed) multi Software Product Lines, hierarchical product lines, or staged configuration and instantiation. The particular focus of EASy-Producer is to support these rather complex concepts in an easy-to use way [EKS11]. This is realized by different views, in particular simplified views for the standard engineering user and expert views for advanced users with comprehensive background knowledge [EEKS14]. The foundation for the current version of EASy-Producer was developed in FP7 INDENICA[14], while other projects such as the nationally funded[15] project ScaleLog allowed for extension, maintenance and validation. EASy-Producer is open source software under Apache 2 license.

Within the range of activities to be carried out in product line engineering, EASy-Producer provides flexible techniques for:

- Variability modeling, i.e., specifying the configuration options as well as dependencies among them in terms of configuration constraints. This is mainly done in IVML, the INDENICA Variability Modeling Language [IND+12, IVML], a powerful language, which provides concepts for developing SPLs and DSPLs.
- The configuration of a particular system based on a given configuration meta model (the variability model). While this can be done in IVML (expert view), EASy-Producer offers simplified graphical editors to describe and validate a configuration. As part of this, a sophisticated reasoning mechanism is used to validate a configuration but also, if possible, to derive values via dependencies.
- Turning a configuration into artifacts of the configured product, e.g., configuration files or adapted source code. In contrast to many (research) approaches, EASy-producer allows to specify this product instantiation (sometimes also called product instantiation) using the Variability Instantiation Language (VIL) [VIL].

Basically, EASy-Producer is an Eclipse plugin and designed to support software product line engineers. An illustration of the high-level logical architecture of EASy-Producer is depicted in Figure 3. In addition, EASy-Producer can be used as a

[13] EASy stands for Engineering Adaptive Systems
[14] www.indenica.eu
[15] Funded by the German Ministry for Economics and Energy (former BMWi)

standalone tool, e.g., to integrate with an existing build process, or, as an even smaller version, as a runtime library. This is achieved in the EASy-Producer architecture by strictly separating those parts that require full Eclipse support (workspaces, user interfaces, JFace as shown in the upper part of Figure 3) from the parts can perform their work based on some runtime classes of Eclipse (lower part of Figure 3). In particular, this separation is supported by the realization of the three languages (IVML, VIL, VTL), where the underlying framework for domain-specific languages xText[16] generates the language infrastructures in terms of an Eclipse-dependent and an Eclipse-independent part. Moreover, the standalone version as well as the runtime libraries of EASy-Producer can even work without a (full) Eclipse/OSGi framework.

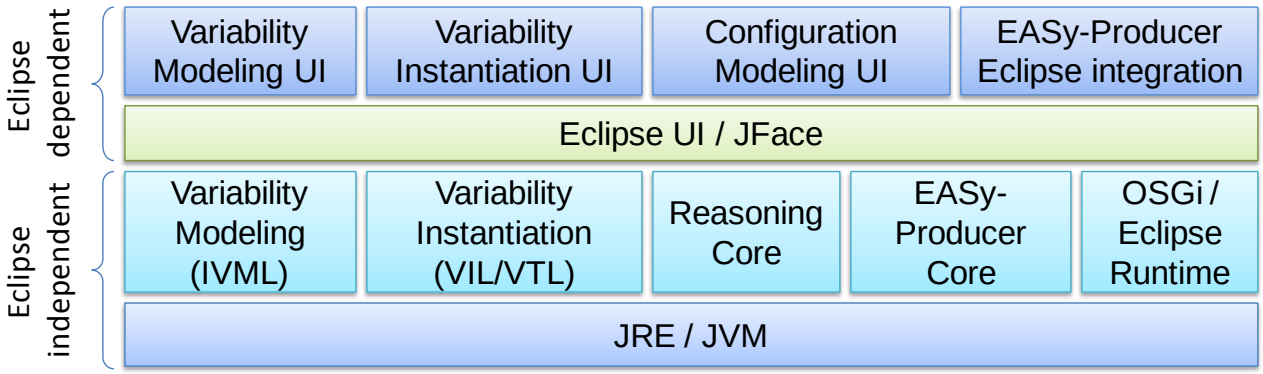

Figure 3: Logical architecture of EASy-Producer.

Several further tools have been developed by the SPLE community (see e.g., [ALR14]), among them commercial tools such as pure::variants [B13] or BigLever Gears [K08] as well as a large set of research tools, such as DOPLER [RHE+10] or FMP [CHE05]. However, tools that can support capabilities required by QualiMaster are either prohibitively expensive or not available to the public (DOPLER), while other relevant tools typically focus on one aspect of SPLE, mostly variability modeling, and, thus, do not support flexible instantiation. Furthermore, using a tool developed by one of the QualiMaster partners eases integration and ensures support and maintenance.

In QualiMaster, the Eclipse version of EASy-Producer will mainly be used by SUH in order to provide the variability model and the product derivation specification for the QualiMaster platform. In order to ease the configuration of the QualiMaster platform for domain experts, specific application frontends on top of the standalone version of EASy-Producer (the Configuration Core layer of the QualiMaster application) will be developed and connected to the repositories. In addition, the

[16] www.eclipse.org/Xtext/

runtime version of EASy-Producer will be used for enacting adaptations through the Coordination Layer and in the Adaptation Layer, in particular for constraint checking.

3.5 Monitoring Resource Consumption

SPASS-meter[17] [ES14, ES12] is a flexible resource monitoring framework for Java programs. It monitors the execution time, memory consumption, system load, file as well as network transfer of a system under monitoring (SUM), and provides access to the collected information at runtime. It can also observe the state of individual heap variables. Therefore, SPASS-meter relies on program instrumentation, i.e., the insertion of additional code into the SUM in order to gain access to relevant low-level information. In contrast to related monitoring approaches such as Kieker [RHM+08], Opencore [JINS] or the built-in tools of Java (please refer to [ES14] for a detailed analysis), SPASS-meter enables **monitoring of user-defined program units**, such as individual components, or services. In the QualiMaster infrastructure, this specific capability enables us to access the resource consumption information of individual data processing algorithms, a prerequisite for the adaptive execution of data processing pipelines. SPASS-meter was partially developed by FP7 INDENICA and is open source software under Apache 2 license.

However, providing information on individual parts of system requires some form of runtime program analysis, as the observed resource consumption must be collected and aggregated while the SUM is running side-by-side (as illustrated in Figure 4). To cope with this potential overhead, SPASS-meter can be configured in a flexible manner, e.g., which resources shall be collected on which part as well as whether parts shall be monitored in isolation or including their dependent libraries. Furthermore, SPASS-meter can dynamically trace execution paths in order to focus the instrumentation on really relevant parts. However, runtime program analysis itself is a resource consuming task and, thus, may lead to a certain overhead. In experimental analyses [ES14, ES12], we have shown that the runtime overhead of SPASS-meter using the SPECjvm2008 benchmark suite [SPEC] as SUM in comparison to Kieker and OpenCore is good (less than 3% processing power overhead and less than 0.5% memory overhead).

[17] SPASS is the acronym for Simplifying the develoPment of Adaptive Software Systems and SPASS-meter is one of the foundational building blocks of our work on that topic. In German, the term “Spass” means “fun” and points to the tons of fun we had while realizing this tool.

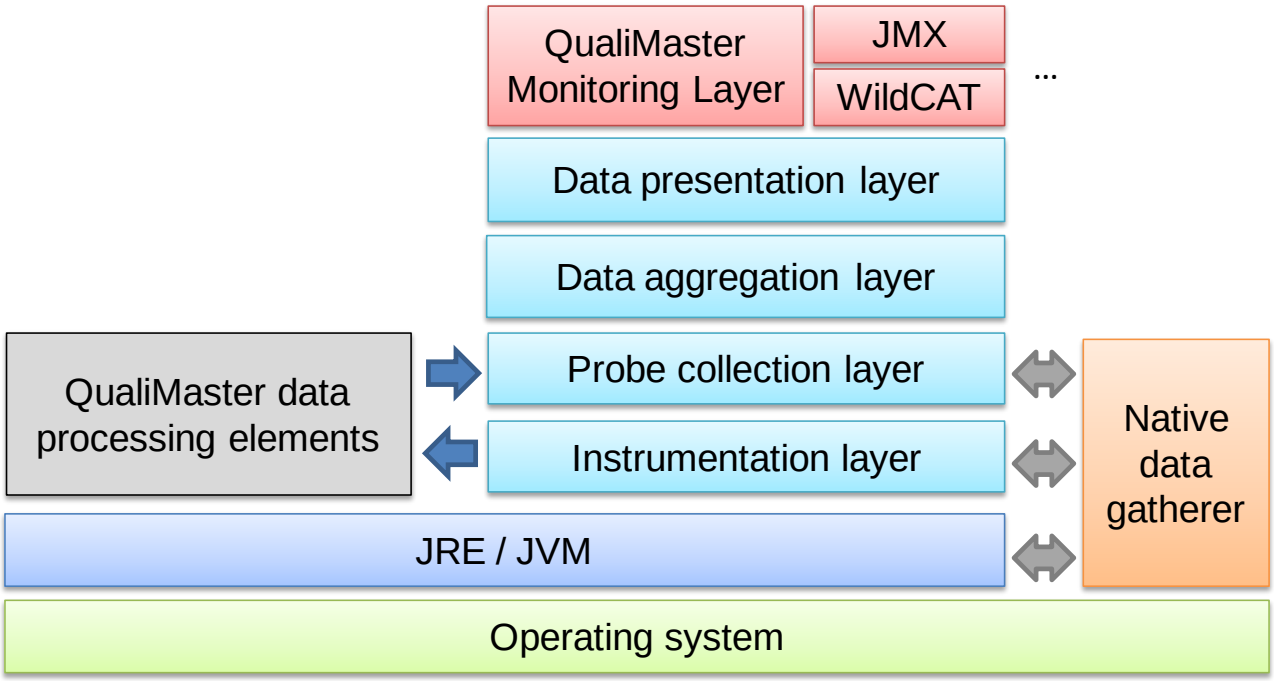

Figure 4: Architecture overview of SPASS-meter.

In the QualiMaster infrastructure, we will focus our work on SPASS-meter for monitoring the real-time data processing algorithms as well as for dynamically changing the monitoring / instrumentation on request at runtime. Therefore, SPASS-meter will be optimized wherever needed in order to comply with the real-time data processing requirements of QualiMaster. Furthermore, it will be integrated as a mediator between the systems in the Execution Layer (to be instrumented) and the Monitoring Layer. The integration with the Monitoring Layer will happen as one of the data presentation layer plugins on top of the strictly layered SPASS-meter architecture as shown in Figure 4.

4 Infrastructure Layers and Tools

In this section we now describe the layers and tools that compose the QualiMaster infrastructure. We group them per lifecycle phase, thus we start with the tools used during the configuration (Section 4.1), continue with start-up time (Section 4.2), and finally describe the layers used during runtime (Section 4.3).

4.1 Configuration

During the configuration phase, the QualiMaster infrastructure is used to define basic settings for the QualiMaster platform, i.e., the definitions of the pipelines to be executed as well as the adaptation space for the pipelines is determined. The related information is collected by three (conceptual) tools, stored in a configuration model and used to derive / instantiate the specific QualiMaster platform. During platform derivation, the execution systems are tailored for optimized execution of the configured pipelines. Further, the monitoring of the execution system is enabled. Ultimately, software artifacts are instantiated (generated or modified), which represent the actual execution, such as the pipeline representation, e.g., for Storm the topology definition as well as the Storm specific adaptive processing elements (bolts) linking to the algorithm families.

4.1.1 Adaptation management tool

In the configuration phase of the QualiMaster, the Adaptation Management Tool (AM tool) provides a user interface and related tooling on top of the Configuration Core to define and specify the adaptive behavior of the QualiMaster infrastructure and the pipelines to be executed. It relates to the implementation of the use cases of the Adaptation Manager (see D1.1, Section 5.2), which defines the quality characteristics of processing elements (UC-AM1) and pipelines (UC-AM2) as well as reactive, proactive, and reflective adaptation rules (UC-AM3 and UC-AM4). Furthermore, it will provide support for monitoring the execution of adaptation rules (UC-AM5), i.e., information and functionality that might also be reused in an overall administrative dashboard. In particular, the adaptation properties and specifications by the AM tool will define the acting behavior of the Adaptation Layer at runtime.

The AM tool uses the interfaces of the common Configuration Core in order to configure the quality characteristics of the processing elements, the pipelines, and the (reactive, proactive, and reflective) adaption rules. From the Configuration Core, the AM tool will derive the Configuration Meta Model as well as the functionality to validate the settings. The configuration results will be stored in the counterpart repository (Processing Element Repository or Pipeline Repository), which will keep the information for other configuration tools and the pipeline instantiation and support the adaptation modifications at runtime.

4.1.2 Pipeline Configuration tool

The Pipeline Configuration tool (PC tool) is used by the Pipeline Designer for managing the QualiMaster pipelines. This tool includes functionally for defining a new pipeline, modifying an already created pipeline, and deleting a pipeline (see D1.1 and use cases UC-PD1, UC-PD2, and UC-PD3). The information related to pipelines, such as the data flow, will be maintained in the Pipeline Repository.

Note that a pipeline is composed by processing elements. Each element will be executed by the Execution layer and not necessarily by the same execution system. For example, we might have elements in pipelines that are defined for being executed by the reconfigurable hardware and other elements on Apache Storm.

4.1.3 Platform Administration tool

The Platform Administration tool (PA tool) provides the capabilities for setting-up, installing, and managing the QualiMaster infrastructure. In the following, we list the key functionalities of the PA tool, which are derived from the use cases of the Platform Administrator as described in Section 5.3 of D1.1. In more detail, the PA tool will offer the following functionalities:

- *Administration of the platform quality parameters:* The PA tool provides an interface for defining and modifying the (low-level) quality parameters for the platform and the methods to measure them. Example parameters are the platform throughput (in terms of processed tuples/sec) or the resources consumed (in terms of memory and CPU resources). This part of the PA tool implements use cases UC-PA-1 and UC-PA-2.
- *Administration of the data processing elements:* The PA tool provides an interface to the QualiMaster repository of data processing elements. The interface will enable the platform administrator to add, modify and remove data processing elements from/into the repository. This involves data processing elements that are implemented on reconfigurable hardware as well. These functionalities realize the use cases UC-PA-3, UC-PA-4, UC-PA-5 and UC-PA-6.
- *Configuration of the platform for software and hardware-based execution:* The PA tool provides the capabilities for configuring the platform for a certain execution environment or when new (reconfigurable) hardware is made available. This implements the use cases UC-PA-10 and UC-PA-11.
- *Configuration of the pipeline sources and sinks:* Using the PA tool, the platform administrator will be able to specify the technical information required for the sources and sinks of the pipelines, such as IP addresses, access credentials etc. The PA interface enables the administrator to define

new sources and sinks, modify existing ones or remove them. This functionality implements use cases UC-PA-7.

4.1.4 Configuration Core

The Configuration Core provides the basic capabilities for flexibly customizing, tailoring, and instantiating the QualiMaster platform. Basically, this consists of the EASy-Producer tool (see Section 3.4). Among a wide range of interfaces of EASy-Producer, the Configuration Core will offer mechanisms to load the QualiMaster configuration meta model in terms of IVML, to load and store configurations, to validate configurations and to reason about them and to instantiate a given configuration using the VIL instantiation specification for QualiMaster. The configuration meta model as well as the instantiation specification are considered to be part of the Configuration Core, but they can be adapted on need, in particular by very experienced users.

4.2 Start-up time

During the startup of an instantiated QualiMaster platform, the required resources are allocated and prepared for the execution. In particular, the reconfigurable hardware is configured by downloading FPGA configuration files. Then all execution systems are powered up, i.e., the startup of Apache Storm and Hadoop. Thereby, the systems may be instrumented as needed, i.e., monitoring probes are inserted in order to inform the Monitoring Layer about resource consumption. Furthermore, additional Monitoring modules connect to the Monitoring Layer. Finally, the initial algorithms to be used for execution are selected and algorithm parameters are determined if needed by the Adaptation Layer. The process of starting up the QualiMaster platform and enabling it to perform its designated processing tasks as configured in the pipeline definition (stored in the pipeline repository) is coordinated by the Execution Layer.

4.3 Runtime

The processing of the streaming data, i.e., the execution of the pipelines for the financial analysis, is performed by the QualiMaster infrastructure during the runtime phase. The performed processing is either manual that performs the enactment of manual decisions/actions suggested by the reflective adaptation but after manual approval, or autonomous that performs the enactment of adaptation actions determined by the reactive, proactive, or (automatic) reflective adaptation.

This phase uses a small set of layers, and more specifically: the **Data Management Layer** for handling the input and resulted data, the **Execution Layer** for invoking the systems that perform specific processing tasks, the **Monitoring Layer** for colleting statistics related to the processing, the **Adaptation Layer** for

making adaptive decisions of individual pipelines by analyzing the statistics of the monitoring layer**,** and the **Coordination Layer** that enacts the actions that take place by the adaptation layer. In the following paragraphs, we provide detailed description for each of the layers used during the runtime phase.

4.3.1 Data Management Layer

The Data Management Layer is responsible for handling the QualiMaster data. This includes the data that are given by the applications as well as the results that are generated by the processing performed by the infrastructure. As illustrated in Table 1, in the QualiMaster application cases the QualiMaster infrastructure will work with data from stock markets as well as data from Social Web. In both cases, the data can be either real-time or historical.

Data Source			
(a) Stock Markets		**(b) Social Web**	
Type		**Type**	
Real-time	Historical	Real-time	Historical

Table 1: Categorization of the data given to the Data Management Layer.

Furthermore, intermediate or final processing results may have to be stored as (updated) historical data to enable future processing. Management of historical data is required by the QualiMaster application use cases (see deliverable D1.1).

In more detail, the Data Management Layer will distinguish between:

- Raw input real time streams that shall be processed by a data analysis pipeline. The Data Management Layer will provide an interface for managing the sources of the data streams (e.g. subscription/unsubscription to online sources of financial and web data streams). Similarly, the interface of this layer will enable managing the sinks of the executed pipelines (i.e. for the output streams). In addition, some applications might require the storage of the raw data coming from the real-time streams (e.g. for later use as historical data or for batch processing based data analysis). However, storing all information in a data stream may not be feasible, e.g., due to the volume of the data stream or even due to legal or licensing issues. Further, enabling the data analysis pipelines to decide how and what to store from an input data stream may easily lead to repeated storage, in particular, if multiple pipelines process the same input data. Thus, the Data Management Layer will handle this in a centralized way based on a global configuration of the storage for a data source, e.g., whether the Data Management Layer shall

store information at all or which information shall be stored where and for how long.

- Historical data input is already available in persistent storage, either within the QualiMaster infrastructure or accessible through the integration of external storages such as the historical financial data archive of SPRING. The Data Management Layer will provide an interface for accessing this persistent storage. Here, the pipeline designer specifies as part of the pipeline design how to access the data, e.g., through a specific data processing element (which accesses the data through the Data Management Layer) or in terms of the API of the Data Management Layer.
- Processed data is data generated by a data analysis pipeline during execution. Processed data may be intermediary data or the final output of a data analysis pipeline. Here, also the pipeline designer specifies where within a data analysis pipeline which data shall be stored in which location, e.g., in terms of a generic data processing element that passes the related data tuples to the Data Management Layer.

4.3.2 Execution Layer

The purpose of the Execution Layer is to provide common access mechanisms to the systems we included in the QualiMaster infrastructure for performing specific processing tasks. Given the user requirements presented in deliverable D1.1 and the detailed discussions in Section 3, we consider the following three systems:

- Storm for the execution of real-time stream processing topologies
- Hadoop for the processing of historical mass data in a batch manner
- Reconfigurable hardware for the execution of hardware-based algorithms

The Execution Layer will receive the requests for these systems, send them for execution. In case there is a failure in one of the systems, the layer is responsible for notifying the other layers in order to address the issue. Having such an execution layer in the QualiMaster infrastructure also allows us to easily incorporate additional systems. Incorporating additional systems might be, for instance, required for satisfying a broader set of requirements or for efficiently executing specific algorithm parts. In particular, such a form of extension is supported by the extensible specification of the configuration and the instantiation of the QualiMaster infrastructure (see Section 3.4).

4.3.3 Monitoring Layer

The Monitoring Layer provides a common interface to all runtime readings taken from the Execution Layer such as actual quality properties and execution state and provides this information to upper level QualiMaster runtime layers, in particular the Adaptation Layer. As a core part of its tasks, it collects and aggregates information from the different systems in the execution layer in a distributed fashion.

The Monitoring Layer will provide the following type of interfaces:

- Input/notification interface for monitoring information obtained from the Execution Layer, and
- Output interface for observing layers to be informed about the (aggregation of the) actual status such as the Adaptation layer.

Basically, the input/notification interface will be used to collect information in three distinct ways, namely through:

1. Instrumentation, i.e., additional notification calls inserted into the execution systems and, in particular, the data processing algorithms. Instrumentation for monitoring the resource consumption of the execution systems, and, in particular, individual algorithms, will be handled by SPASS-meter (see Section 3.5).
2. Explicit notification of monitored information. If required, this will typically happen from source code generated during the instantiation of the QualiMaster infrastructure, i.e., produced by VIL specifications executed by EASy-Producer (see Section 3.4).
3. Generic monitoring providers, i.e., software components using interfaces provided by the respective execution system in order to access the execution status on various levels of granularity, e.g., the Thrift[18] interface of Storm. Fundamentally, the Thrift interface of Storm provides a service to derive the running information about the specific topology (the Storm representation of a QualiMaster pipeline). Thereby, the statistics of the running pipeline on topology level (not on algorithm level) can be detected via the Monitoring Layer.

4.3.4 Coordination Layer

The Coordination Layer will enact the actions that are requested by the adaptation layer. This plays a fundamental role in the correct operation of the QualiMaster system. The specific layer will provide three different interfaces with the other layers:

1. Interface with the Adaptation Layer. This interface will be used for receiving enactment "commands" from the Adaption Layer about the system's pipeline infrastructure. In addition, this interface will be used for informing the Adaptation Layer about the results of previous changes. Last, this interface will also be used for giving information to the Adaptation Layer about any failures or faults discovered during the execution of the algorithms.

[18] https://thrift.apache.org/

2. Interface with the Monitoring Layer. This interface will be used for monitoring of the running processes through the Monitoring Layer. Also, this interface will be used for changing the monitoring priority, when such commands arrive from the Adaptation Layer, e.g. in case that the Adaptation Layer needs to monitor another part of the pipelined system.
3. Interface with the Execution layer. As the Coordination Layer is used for enacting the different processing modules in the pipeline, this layer will be responsible for establishing “pipes” between the blocks that are used for the processing, e.g. the connection between the Storm and the reconfigurable platforms. Also, this interface will be used for fault detections and sending the proper information to the adaptation layer for fault handling.

The described interfaces will be used for the fulfillment of different operations.

1. First, the Coordination Layer will enact the adjustments of the adaptation layer for the data analysis pipelines during the processing runtime. The Coordination Layer will receive some specific commands from the Adaptation Layer and it will translate them into commands for the individual systems in the Execution layer, e.g. Storm, Hadoop, so that to coordinate individual multiple affected systems. Using the Coordination Layer as command mediator enables us to include other or even exchange execution systems if actually required. This process will be based on the runtime version of EASy-producer tool in order to process the adaptation actions described in terms of runtime instantiations.
2. Second, the Coordination Layer will be used for the coordination-communication level between the different software and hardware-based platforms of the system. This system will handle the efficient interconnection among the various pipeline infrastructures following the commands of the Adaptation Layer.
3. Third, the Coordination Layer will be able to monitor the execution of the processing systems through the Monitoring Interface. In more details, this layer will monitor the execution process and inform the Adaptation Layer for any malfunction of the malfunctioned processing module. So, this layer will offer high level fault detection and exception handling.
4. Last, the Coordination Layer will “supervise” the activation and the deactivation of the processing elements. Also, it will be responsible for any monitoring priority changes.

4.3.5 Adaptation Layer

Based on the readings of the quality parameters provided by the Monitoring Layer, the Adaptation Layer analyses this information to make adaptation decisions on individual pipelines as well as across pipelines (cross-pipeline optimization).

Besides the information on quality parameters, those decisions derive from the quality characteristics, the adaptation rules and the pipeline constraints that have been specified by the Adaptation Management Tool (see Section 4.1.1), the Pipeline Design Tool (see Section 4.1.2) or the Platform Administration Tool (see Section 4.1.3), respectively. In this layer, adaptive algorithms configured according to the information collected by the Adaptation Management Tool will be utilized for making adaptation decisions and to enact those decisions via the Coordination Layer. Therefore, the Adaptation Layer will send an enactment request to the Coordination Layer. The Coordination Layer enacts the adaptive decision by translating it into specific commands for the individual systems in the Execution layer (Storm, Hadoop or FPGAs), possibly coordinating individual enactments across multiple affected systems. In addition, the Adaptation Layer also interacts with the Monitoring Layer (via the Coordination Layer) for disabling or enabling the observation of the resources.

5 Current status and Future plans

This section describes the current status of the QualiMaster infrastructure (Section 5.1) along with the prepared execution environment (Section 5.2).

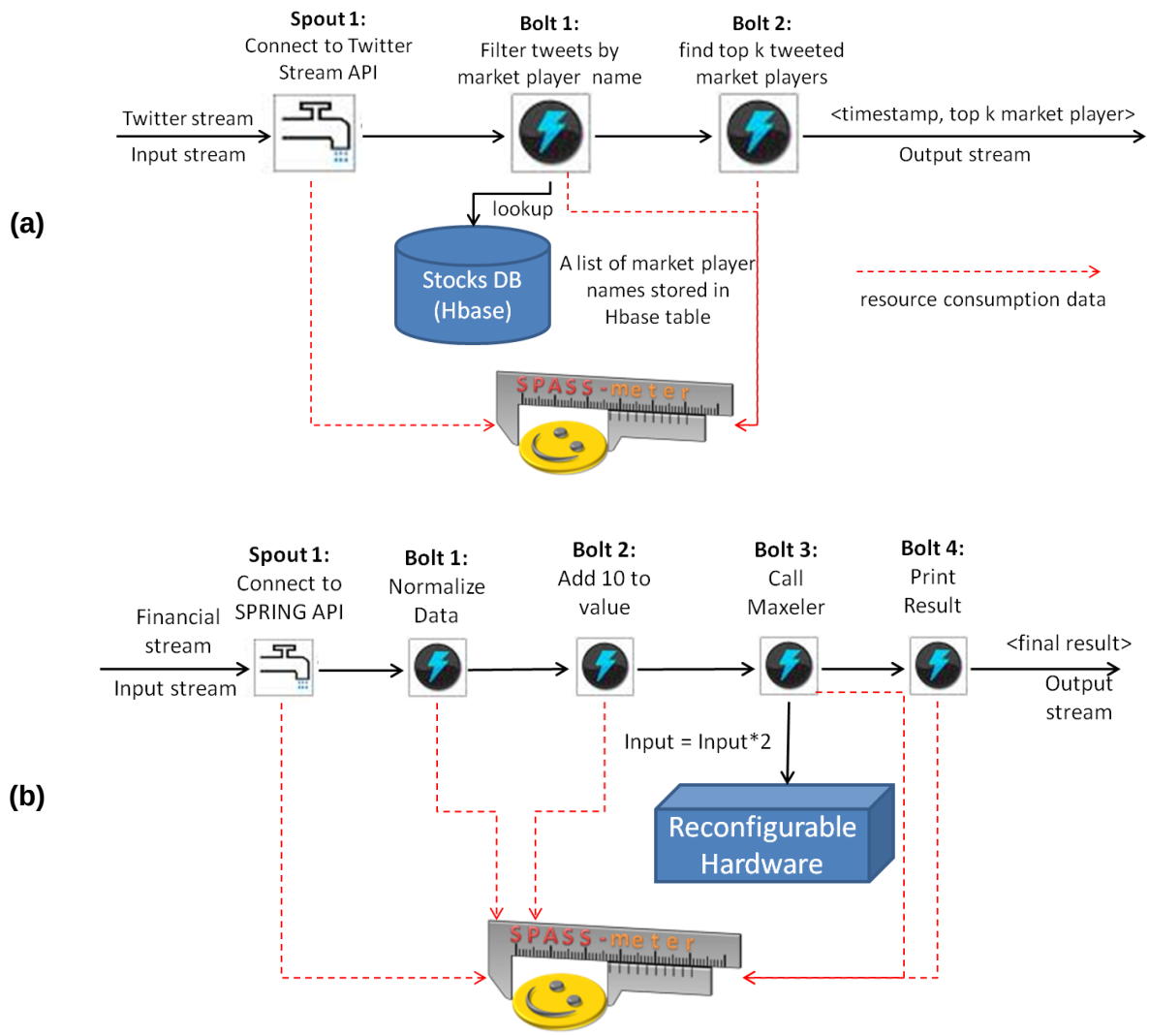

Figure 5: The twitter pipeline (top image) and the financial data pipeline (bottom image) that were used for testing the created and configured infrastructure. SPASS-Meter is used for monitoring the resource consumption of both pipelines.

5.1 Current Status

The goal of WP5 is to define and develop the infrastructure for QualiMaster. This includes the set-up of the overall architecture as well as the actual implementation of the various necessary layers and tools. For the QualiMaster infrastructure we also need to devote effort in the interconnections between the various layers, tools,

and the used external systems, as for example the physical integration of reconfigurable hardware within the Execution layer. In addition to the design and implementation, WP5 needs to work on the setup of the execution environment, the deployment of incremental versions of the infrastructure to the execution environment and the setup of the continuous integration and testing environment.

From the start of the project, the consortium has focused on creating the initial architecture, design, and setup of the QualiMaster infrastructure. As discussed in the previous sections, this included the analysis of the collected requirements (deliverable D1.1), the creation of the QualiMaster infrastructure along with the definition of its layers/tools and a common understanding of the layers/tools as well as the flow between these layers/tools.

The current results of WP5, which we reported in this deliverable, have been used for creating an actual infrastructure. In order to assist the testing of the created and configured infrastructure, and especially of the interaction between the tools and external systems (e.g., Storm, Hadoop, Maxeler nodes, and SPASS-meter), we have defined and used the following two bootstrapping pipelines:

1. **Twitter pipeline**, illustrating the real-time processing of Twitter feeds using Storm in combination with HBase, which computes the top-k of the most frequently tweeted market players. The pipeline connects to Twitter's public Streaming API[19] and receives in real-time the sample stream. The connection to the Streaming API is implemented in Spout 1 of the illustrated Storm topology. Spout 1 then simply emits all the tweets that are received from the Twitter API to the next Bolt in the topology, namely Bolt 1, which in turn filters the tweets stream by looking up in a table of market players that is stored in HBase. Bolt 1 then emits only tweets that mention at least one of the listed marketed players. Bolt 2 receives the filtered stream from Bolt 1 and computes the frequency of the tweets per market player. Bolt 2 then omits a continuous stream of tuples as the final outcome of the pipeline. Each of the tuples in the output stream consists of a timestamp and a list of top-k market players.
2. **Financial data pipeline**, illustrating the connection of the QualiMaster infrastructure to the API of Spring for receiving the real-time streaming data, the use of multiple bolts with simple arithmetic calculations, and the interconnection with reconfigurable hardware. More specifically, Spout 1 performs the connection to the API of SPRING and receives the real-time financial data. Received data go through a series of bolts. Bolt 1 performs normalization and Bolt 2 increases the data value. Bolt 3 receives the filtered

[19] https://stream.twitter.com/1.1/statuses/sample.json

stream from Bolt 2 invokes reconfigurable hardware (i.e., Maxeler dataflow engines), which performs an additional numerical calculation over the data value. The final bolt, i.e., Bolt 4, is responsible for providing a continuous stream of tuples as the final outcome of the pipeline.

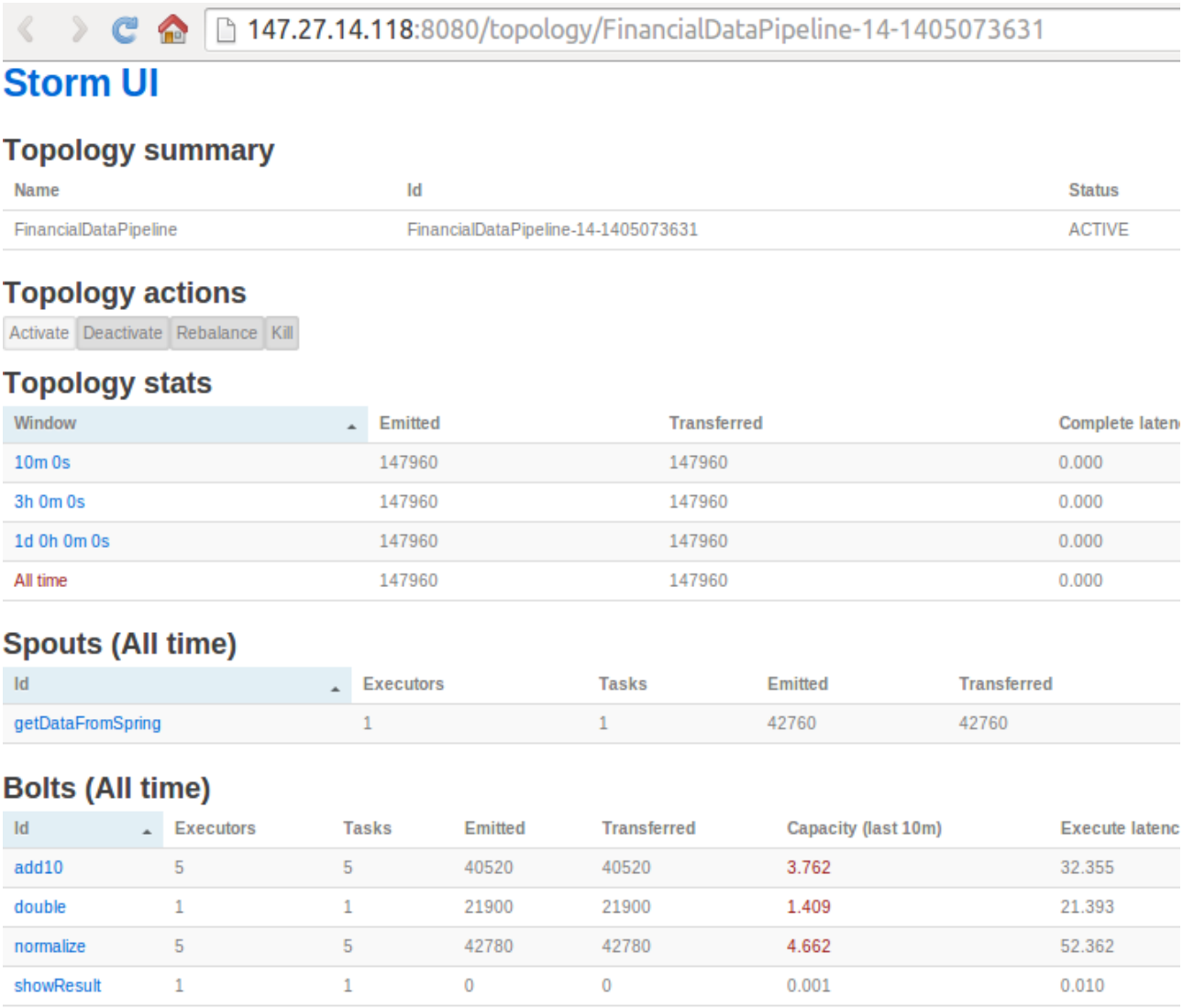

Figure 6: Storm UI showing the execution of the financial data pipeline.

Figure 5 shows an illustration of the twitter pipeline (top image) and of the financial data pipeline (bottom image), which were used for verifying the integration and testing of the created QualiMaster infrastructure. In both pipelines, SPASS-Meter is used for monitoring the execution. Figure 6 shows the execution details for the financial data pipeline and Figure 7 the Twitter pipeline, as provided by Storm UI. More specifically, Storm UI shows that this pipeline contains one spout and four bolts and for each spout/bolt we can see the size of the emitted and transferred information.

In addition to the execution of these two pipelines, initial experiments have been carried out with EASy-Producer in order to validate the required modeling and instantiation capabilities for QualiMaster. Therefore, an initial configuration meta model for EASy-Producer in IVML has been set up which models essential properties of the cluster hardware, the reconfigurable hardware, the processing families and the pipelines. Based on this configuration meta model, an initial platform instantiation process using VIL and VTL has been defined. Finally, an example configuration based on the financial data pipeline has been created. This configuration has been used as input to the initial platform instantiation process and code artifacts matching the manual implementation have been derived successfully. More precisely, the topology (using another naming scheme for the processing elements) as well as stub spouts and bolts were generated. The generation of full code artifacts for spouts and bolts will be available as soon as the detailed architecture of the processing elements is defined (as part of deliverable D5.2). We also tested the Twitter pipeline described above. Also here, the STORM topology can be generated, but the use of HBase is currently not reflected in the configuration model. As historical data will be handled by the Data Management Layer in future, the configuration meta model and the platform instantiation process will be modified as soon as the detailed design of this layer is available. Then, the respective parts of EASy-Producer and a stabilized version of the configuration meta model will be integrated with the QualiMaster infrastructure.

147.27.14.118:8080/topology/TwitterStreamProcessingPipeline-21-1405595672

Storm UI

Topology summary

Name	Id	Status
TwitterStreamProcessingPipeline	TwitterStreamProcessingPipeline-21-1405595672	ACTIVE

Topology actions

Activate Deactivate Rebalance Kill

Topology stats

Window	Emitted	Transferred	Complete latency (ms)
10m 0s	50040	50040	0.000
3h 0m 0s	774280	774280	0.000
1d 0h 0m 0s	774280	774280	0.000
All time	774280	774280	0.000

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Comple
TwitterStreamSpout	2	2	772820	772820	0.000

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)
CountTweetsBolt	1	1	0	0	0.001	3.548
FilterTweetBolt	1	1	1460	1460	0.002	0.018

Figure 7: Storm UI showing the execution of the Twitter pipeline.

For monitoring the execution of the pipelines, we integrated SPASS-meter into the Storm execution system for the infrastructure setup. Therefore, we configured Storm and SPASS-meter so that we are able to generically monitor the resource consumption of all executed Storm spouts and bolts on the distributed worker machines. Currently, the results are recorded into individual log files, one per JVM spawned and utilized by Storm to execute the spouts and bolts defined defined by the pipelines. As soon as the architecture of the Storm processing elements with respect to QualiMaster data sources, data sinks, data management elements and (algorithm family based) data processing elements is detailed, we will adjust the monitoring to focus on the actual algorithms. Furthermore, SPASS-meter will be prepared for the integration and integrated with the Monitoring Layer and the Coordination layer in order to provide resource consumption information at runtime to the QualiMaster infrastructure and to enable adaptive monitoring.

5.2 Execution Environment

We created the execution (and development) environment of QualiMaster on a cluster. We used an existing distribution instead of individually installing each the required systems and manually performing the verification of their interconnection. Thus, we focused only on installing and verifying the required systems that were not included in the specific distribution.

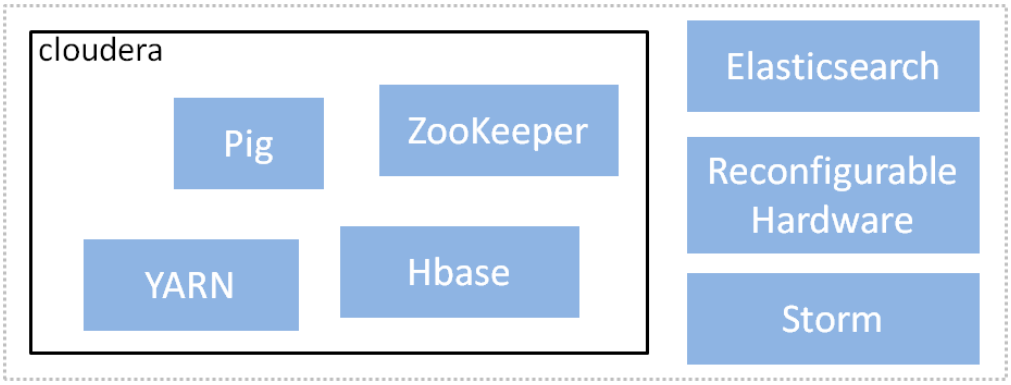

Figure 8: The systems composing the QualiMaster execution environment.

More specifically, we used the Cloudera express distribution[20], version 5.0.2. Cloudera is a popular distribution since it allows the fast and easy installation, configuration, and deployment of applications based on Apache Hadoop and related technologies. There are other distributions that can be used, for example the Hortonworks distribution[21]. We selected the Cloudera distribution because it has been present longer than Hortonworks and has a larger number of customers.

The following list provides the projects incorporated in the Cloudera distribution that are either currently needed by the QualiMaster infrastructure or we have future plan in which these systems might be used:

- ZooKeeper (version 3.4.5)
- Hbase (version 0.96.1.1)
- Pig (version 0.12.0)
- YARN (version 2.3.0)

We also incorporated systems that are required for the QualiMaster Infrastructure and are not included in the Cloudera distribution. These are:

- Apache Storm (version 0.9.1)
- Elasticsearch (version 1.0)

[20] http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html
[21] http://hortonworks.com/

Figure 8 shows the systems composing the QualiMaster execution environment. As shown, in addition to the systems used from the Cloudera distribution, QualiMaster also used Storm, Elasticsearch, and reconfigurable hardware. The latter corresponds to a Maxeler MPC-C Series node and communication with this node is performed using standard TCP sockets.

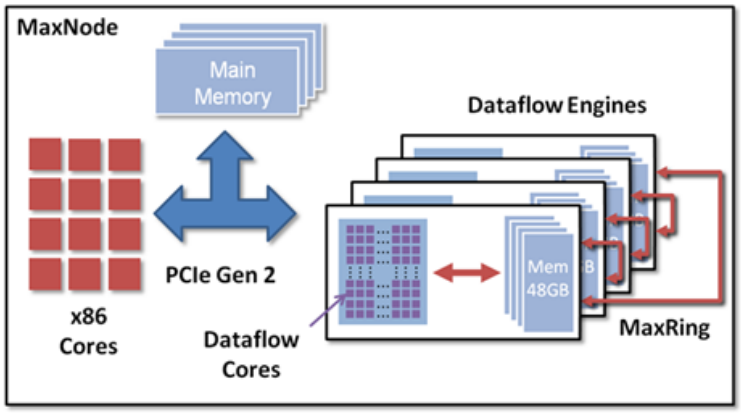

Figure 9: Architecture of MaxNode.

The reconfigurable hardware, i.e., the Maxeler MPC-C Series node, is an independent computer node and every procedure has to setup communication with the machine. Thus, the QualiMaster infrastructure also incorporated and can use the 2 MPC-C series nodes located at the Technical University of Crete. A 1U MPC-C series machine is a server-class HPC system with 12 Intel Xeon CPU cores, 92GB of RAM for the CPUs and 4 dataflow compute engines (using Virtex 6 FPGAs) closely coupled to the CPUs with 24GB of RAM each. Every node can typically provide the computational performance of 20-50 standard x86 servers. Each dataflow engine is connected to the CPUs via PCI Express (2GB/s), and DFEs within the same node are directly connected with the MaxRing interconnect. Figure 9 shows the architecture of the Maxeler Node.

Please note that the execution environment will be further extended in order to allow continuous testing and integration of future algorithms and processes. The corresponding description and discussion will be included in the upcoming WP5 deliverables.

6 Conclusions

In this deliverable, we described the QualiMaster infrastructure in terms of the incorporated layers, tools, and external systems. More specifically, we have provided an overview of the designed QualiMaster infrastructure along with a discussion about the flow between the infrastructure's tools and layers. We have also given a description of our initial execution environment, which we plan to build and extend during the following months. In addition, we provided overviews of the external systems incorporated in QualiMaster and a discussion on the reasons that led to their selection among the other available options. We also provided the details of the infrastructure layers and tools, grouped according to the lifecycle phases they belong to, i.e., configuration, start-up time, or runtime lifecycle phases. Furthermore, we explained the current status of the QualiMaster infrastructure and sketched our plans for the following months.

In the following months, and especially until the end of the year, the consortium has made plans for extending and enhancing the architecture and implementation of the QualiMaster infrastructure. In particular, we plan to work in three main directions. The first direction is to work on the detailed design of the layers and tools that will be incorporated in the QualiMaster infrastructure.

The second direction is starting the integration of the QualiMaster infrastructure. This includes the implementation of the two repositories (i.e., processing elements repository and pipeline repositories) and the implementation of the tools and layers. The goal is to have a first version running by the end of the year.

The third direction is to implement and verify the priority pipeline (Section 1.2). For this direction, we need to have the major tools and layers already implemented and working with the infrastructure, but also additional algorithms and components, such as the algorithm for computing the correlations between multitude market players.

As scheduled in the DoW, this deliverable describes the initial version of the QualiMaster infrastructure. It will be followed by the stabilized design (D5.2, due in month 12), the first version of the integrated infrastructure (D5.3, due in month 21), and the final version of the infrastructure (D5.4, due in month 33). This deliverable as well as the future WP5 deliverables, accompanies the current version of the created QualiMaster infrastructure.

References

[ES14] H. Eichelberger, K. Schmid. Flexible Resource Monitoring of Java Programs, Journal of Software and Systems (93), 2014.

[ALR14] M. Acher, R. E. Lopez-Herrejon, R. Rabiser, A Survey on Teaching of Software Product Lines, In Variability Modelling of Software-intensive Systems (VAMOS), 2014.

[B13] D. Beuche. Modeling and building product lines with pure::variants. In International Software Product Line Conference (SPLC) Workshops, 2013.

[BMK+11] A. Brito, A. Martin, T. Knauth, S. Creutz, D. Becker de Brum, S. Weigert, and C. Fetzer. Scalable and Low-Latency Data Processing with Stream MapReduce. In CloudCom, 2011.

[BRN+13] T. Berger, R. Rublack, D. Nair, J. Atlee, M. Becker, K. Czarnecki, and A. Wasowski. A Survey of Variability Modeling in Industrial Practice. In Variability Modelling of Software-intensive Systems (VaMoS), 2013.

[CHE05] K. Czarnecki, S. Helsen, U. Eisenecker. Staged configuration through specialization and multilevel configuration of feature models, Software Process: Improvement and Practice, vol. 10, issue 2, 2005.

[CN02] P. Clements, L. Northrop. Software Product Lines - Practices and Patterns, Addison Wesley, 2002.

[DPR00] A. Dandalis, V. Prasanna, and J. Rolim. An Adaptive Cryptographic Engine for IPSec Architectures. In Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[DSE98] A. Dollas, E. Sotiriades, A. Emmanouilides. Architecture and Design of GE1, a FCCM for Golomb Ruler Derivation. In Proceedings, 6th International IEEE Symposium on FPGA's for Custom Computing Machines, 1998.

[EEKS14] H. Eichelberger, S. El-Sharkawy, C. Kröher, K. Schmid, EASy-Producer - Product Line Development for Variant-Rich Ecosystems, International Software Product Line Conference (SPLC), 2014 (accepted).

[EKS11] S. El-Sharkawy, C. Kröher, and K. Schmid. Supporting Heterogeneous Compositional Multi Software Product Lines. In International Software Product Line Conference (SPLC), Volume 2, 2011.

[EP00] A. Elbirt, and C. Paar. An FPGA implementation and performance evaluation of the Serpent block cipher. In Proceedings of the Eighth ACM/SIGDA International Symposium on FPGAs 2000.

[ES12] H. Eichelberger, K. Schmid. Erhebung von Produkt-Laufzeit-Metriken: Ein Vergleich mit dem SPASS-Meter-Werkzeug. In Proceedings of the DASMA Metrik Kongress (MetriKon), 2012 (in German).

[HHPS08] S. Hallsteinsen, M. Hinchey, S. Park, K. Schmid, Dynamic Software Product Lines, IEEE Computer (41), 4, 2008.

[HLA98] R. Hudson, D. Lehn, P. Athanas. A run-time reconfigurable engine for image interpolation. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1998.

[HSM00] W. Huang, N. Saxena, and E. McCluskey. A Reliable LZ Data Compressor on Reconfigurable Coprocessors. In Proceedings of the 8th International IEEE Symposium on FPGA's for Custom Computing Machines, 2000.

[IND+12] INDENICA project consortium. Open Variability Modeling Approach for Service Ecosystems, Deliverable D2.1, 2012, via http://www.indenica.eu [validate: June

2014]

[IVML] IVML team, IVML Language Specification, http://projects.sse.uni-hildesheim.de/easy/docs/ivml_spec.pdf [validated: June 2014]

[JIns] JInspired, Opencore – Enterprise Software Resource Metering & Application Performance Metric Monitoring, Available at: http://opencore.jinspired.com/

[K08] C. W. Krueger. The BigLever Software Gears Unified Software Product Line Engineering Framework. In International Software Product Line Conference (SPLC), 2008.

[KBD03] C. Kachris, N. Bourbakis, A. Dollas. A Reconfigurable Logic-Based Processor for the SCAN Image and Video Encryption Algorithm. International Journal of Parallel Programming 31(6), 2003.

[KM00] H. Kim and W. Mangione-Smith. Factoring large numbers with programmable hardware. In Proceedings of the Eighth ACM/SIGDA International Symposium on FPGAs ACM/SIGDA International Symposium on FPGAs, 2000.

[LC97] Y. Li and W. Chu. Implementation of single precision floating point square root on FPGAs. In Proceedings of FCCM, 1997.

[LCTL00] M. M Leong, O. Cheung, K. Tsoi, and P. Leong. A Bit-Serial Implementation of the International Data Encryption Algorithm IDEA. In Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[LLR+99] W. Luk, T. Lee, J. Rice, N. Shirazi, P. Cheung. Reconfigurable Computing for Augmented Reality. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1999.

[LMWL00] K. Leung, K. Ma, W. Wong, and P. Leong. FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor. In Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines 2000.

[LSR07] F. van der Linden, K. Schmid, and E. Rommes. Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering, Springer, 2007.

[M98] M. Shand, L. Moll. Hardware/software integration in solar polarimetry. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1998.

[MFB11] A. Martin, C. Fetzer, and A. Brito. Active Replication at (Almost) No Cost. In SRDS, 2011.

[NRNK10] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed Stream Computing Platform. In International Conference on Data Mining Workshops (ICDMW), 2010.

[P00] C. Patterson. High Performance DES Encryption in Virtex(tm) FPGAs Using Jbits(tm). In Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[PBL05] K. Pohl, G. Böckle, F. van der Linden. Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, 2005.

[PR97] C. Paar and M. Rosner. Comparison of arithmetic architectures for Reed-Solomon decoders in reconfigurable hardware. In Proceedings of FCCM, 1997.

[PVH99] M. Piacentino, G. VanderWal, M. Hansen. Reconfigurable Elements for a Video Pipeline Processor. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1999.

[RH97] M. M. Rencher, B. Hutchings. Automated Target Recognition on Splash 2. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1997.

[RHE+10] R. Rabiser, W. Heider, C. Elsner, M. Lehofer, P. Grünbacher, and C. Schwanninger. A flexible approach for generating product-specific documents in product lines. In International Conference on Software Product Lines (SPLC), 2010.

[RHM+08] M. Rohr, A. van Hoorn, J. Matevska, N. Sommer, L. Stoever, S. Giesecke, W. Hasselbring. Kieker: Continuous monitoring and on demand visualization of Java software behavior, Proceedings of the IASTED International Conference on Software Engineering (SE), 2008.

[SAA95] N. Shirazi, P. Athanas, and A. Abbott. Implementation of a 2-D Fast Fourier Transform on an FPGA-Based Custom Computing Machine, 1995.

[SDA00] E. Sotiriades, A. Dollas, and P. Athanas. Hardware-Software Codesign and Parallel Implementation of a Golomb Ruler Derivation Engine. In Proceedings 8th International IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[SDD00] P. Stogiannos, A. Dollas, V. Digalakis. A Configurable Logic Based Architecture for Real-Time Continuous Speech Recognition Using Hidden Markov Models. VLSI Signal Processing 24(2-3), 2000.

[SPEC] SPEC corporation, SPECjvm2008 benchmark suite, http://www.spec.org/jvm2008/

[VIL] VIL team, VIL Language Specification, http://projects.sse.uni-hildesheim.de/easy/docs/vil_spec.pdf [validated: June 2014]

[WL99] M. Weinhardt and W. Luk. Pipeline Vectorization for Reconfigurable Systems. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1999.