DOI 10.12694/scpe.v16i1.1062
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 85–101. http://www.scpe.org © 2015 SCPE

EXTENDING XNAT TOWARDS A CLOUD-BASED QUALITY ASSESSMENT PLATFORM FOR RETINAL OPTICAL COHERENCE TOMOGRAPHIES[*]

CHRISTOPH JANSEN[†] MAXIMILIAN BEIER[†] MICHAEL WITT[†] JIE WU[‡] AND DAGMAR KREFTING[†]

Abstract. Neuroscientific research is increasingly based on image analysis methods. Among them, optical coherence tomography (OCT) allows for non-invasive visualisation of anatomical structures on a micrometer scale. The platform presented in this paper is designed for automatic quality assessment of retinal OCTs. It extends the image management platform XNAT by services to calculate and store quality measures. It is also extensible regarding new quality measure algorithms, allowing the developer to upload, compile and test code for the system's architecture. The processing tools are provided as a cloud-based service employing OpenStack. Different approaches using fully equipped virtual machines and Docker containers are investigated and compared regarding security and performance aspects.

Key words: medical imaging, cloud, SaaS, IaaS, XNAT, OCT, OpenStack, Docker

AMS subject classifications. 68M14, 68M20, 68U10, 68U35

1. Introduction.

Modern neuroscience research is increasingly using imaging techniques such as magnetic resonance imaging (MRI) and computed tomography (CT). Among these techniques, optical coherence tomography (OCT) utilizes the characteristic refractivity of different tissue for non-invasive visualisation of anatomical structures on micrometer scale. Becoming a standard diagnostic tool in ophthalmology, it is of rising interest for investigation of multiple sclerosis, as correlation between retinal changes and the patient's symptoms could be found [1, 2].

1.1. Image Quality.

Virtually all data processing relies on sufficient data quality, but few tools inherently validate the quality of the input data. In today's complex image processing, as for example the FreeSurfer pipeline for segmentation of anatomical brain MRI, it is difficult to manually evaluate in advance, if a certain image's quality is sufficient for the envisioned processing [3]. Information about input data requirements are often only given in the software documentation and it is left to the user to use appropriate input data [4]. Today this issue is addressed by offering a visual inspection and interactive correction of intermediate results. This is a feasible approach if there are few datasets, but with increasing numbers of datasets and patients in clinical research, manual processing of the images becomes hardly manageable. Furthermore, the evaluation of the appropriateness of an image for a certain analysis method may require a high level of expertise and experience. Therefore, automatic in-advance quality validation, as known from commercial online photo print services, would help the researcher to avoid quality-related result errors. In this paper, a platform specifically designed for *Quality Management for Retinal Optical Coherence Tomography* (QMROCT) is presented. At least for the most commonly used type of retinal OCT scan, the so-called ring scan, quality criteria are specified, known as OSCAR-IB criteria [5]. The criteria are categorized into the groups "(O) obvious problems, (S) poor signal strength, (C) centration of scan, (A) algorithm failure, (R) retinal pathology other than MS related, (I) illumination and (B) beam placement". Some of these features – for example the signal strength – are easily evaluated automatically, while others, such as the correct position, might require more complex image analysis including certain segmentation and image registration tasks itself.

1.2. Research Platform.

Medical image data management for research requires further functionalities than classical picture archiving and communication systems (PACS) used in daily care. Additional features are for example the grouping of data into different clinical trials, storage and management of non-DICOM data and fine-grained access control [6]. An increasingly used open source platform supporting such features is the "Extensible Neuroimaging Archive Toolkit" (XNAT) [7]. In conjunction with several collaborative groups

[*] The presented work is supported by the German Federal Ministry of Education and Research (Somnonetz-01EZ1132), the German Federal Ministry of Economic Affairs and Energy (Macchiato), the Berlin Institute for Applied Research (QM ROCT), and the ITEA2-programme (EASI-CLOUDS).

[†]University of Applied Sciences Berlin Germany, Email: c.jansen@student.htw-berlin.de
[‡]TMF e.V. Berlin, Germany

around the world, the INCF Task Force has also started work on several tools to ease and eventually automate the practice of data sharing in field of neuroimaging [8]. The goal is to allow researchers to easily share raw, processed, and derived neuroimaging data and improving the reproducibility of neuroimaging studies.

1.3. Cloud infrastructures for medical imaging. Medical image processing methods for post-processing and analysis to study the normal and pathological brain structure and function often have high demands on computing power and memory. The resulting requirements on data and computing availability and reliability are specifically addressed by cloud infrastructures. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [14]. Common cloud platforms are Microsoft Azure [10], Amazon EC2 [11], Google Cloud Platform [12] and the open source platform OpenStack [13]. Recently, research projects have been started to implement cloud-based neuroimaging applications. The German part of the EASI-CLOUDS [20] project implements a cloud-based neuroradiologic analysis platform to process compute-intensive clinical examinations. The project focuses on Service Oriented Architecture (SOA) and Service Level Agreement (SLA) management for several applications. The NeCTAR group is working on providing cloud-based image analysis and processing software packages [21] for research communities via remotely accessible user-interfaces. This project focuses on processing large image data sets in a cloud environment. To control the processing, a workflow management is integrated and used as the user interface. To improve performance in large medical image access and processing, a cloud-based multi-agent system is built for image retrieval by Alonso-Calvo and others [22]. Here, single images are divided so that the resultant sub-images are stored and managed separately by different agents distributed within the cloud. Da Mota and co-workers [23] are using Microsoft Azure to apply machine learning algorithms to functional Magnetic Resonance Imaging (fMRI) data in a MapReduce approach. They are mainly focusing on algorithms, as well as efficient and stable processing. R & D Cloud CEIB [24] is a bioimaging project developing a modular cloud software for classification purposes. This team also extends XNAT for their needs in bioimaging by using its generic DICOM and XML data types. Furthermore, many solutions developed for grid-based neuroimage processing can be adopted to cloud infrastructures. For example, web-based Grid portals such as the Charité Grid portal [15] and the WS-PGRADE portal [16] provide components and services (e.g. Grid Security Infrastructure (GSI) [17] based authentication and authorization, workflow management, pseudonymisation service especially for patient data and secure data transfer) which can also be used in cloud-based neuroimage processing. As an additional feature for applications with user interaction, the visualisation of intermediate results can also be integrated into neuroimage applications [19].

2. Requirements.

The overall goal of the presented platform is to facilitate state-of-the-art image quality assessment. This implies dynamically growing image data and quality measures. New images are created by the scientists performing clinical trials. The image data is assumed to be acquired in intermittent bursts, whenever a clinical trial is in its patient recruiting phase. In contrast, new quality measures are expected to be developed continuously but in a much larger time-scale. From the scientist's perspective, the system must provide the following features:

1. Whenever an image is stored in the platform, the currently available quality measures are calculated.
2. The calculated quality measures are accessible in the context of the image.
3. For manual inspection, a visual representation of the calculated quality features is accessible in the context of the images.
4. Whenever a new quality measure is integrated into the system, it is automatically calculated for all stored images.
5. Whenever a quality measure is updated, the new version is calculated for all images, and the former values are archived.

2.1. Users. Two abstract types of users are identified: The *clinical researcher*, who manages his or her medical images and studies within the platform; and the *computer scientist*, who develops new automated quality measure methods and integrates them into the platform. All interfaces of the platform to the clinical researcher should be integrated into his or her daily working environment, whereas the algorithm developer must not have access to the medical data stored within the system, except authorized test data. It is possible to subdivide the two types further, for example by setting up different projects for different medical teams in the research platform. The development process of the algorithms themselves is not part of the platform. Developers prefer to use their own personalized environment for code development, in our case Matlab. It implies that an interface to integrate new methods must be provided for the developer. In particular, the developer needs to be able to control the reliability of the code on the platform, so he or she must be able to run the code on the platform with test data and confirm the intended behaviour of the integrated software.

2.2. Resources. Regarding resource consumption, we expect constantly growing storage and strongly varying computing power requirements. Whenever new data or new processing methods are integrated into the platform, image processing is necessary. Each new image will be analysed with all currently available methods and each new algorithm will be applied to all currently available images. Such a scenario with strongly varying resource requirements might strongly benefit from scalable computing resources, as provided by cloud infrastructures. As the processing of the individual images and algorithms are independent from each other, they can be executed simultaneously in a distributed environment. As a possible drawback, data transfer might be a significant factor for the total processing time in distributed environments, as image processing is in most cases data-intensive. So besides scalable computing resources, broadband network capabilities might also be crucial for performance earnings.

2.3. Security. There are several security requirements regarding the envisioned platform, from medical data protection to malicious code execution.

Medical data protection. As medical images are stored and processed, personal data protection must be ensured. While the retina of the eye is known to be unique in every human individual, the OCT data implicitly contain reidentification potential. On the other hand, reidentification based on the OCT data is only possible, if the attacker already has access to the subject's data. In contrast to e.g. reconstructed 3D-MRT of a head, the identification features of the OCT are not observable by everyone. The envisioned platform will be used by a clinical trials unit of the university clinics, which implies, that the data is specifically acquired for study purposes and not for patient care. The image metadata do not contain the patient's name but a pseudonym that is managed locally within the clinical trials unit. The only identifying information found in the metadata is the patient's birthday, which is automatically changed to the year of birth in a preprocessing step before entering the platform. We assume that strong data protection measures like file-based encryption are not required in the current setting.

Confidentiality of the research topic. As neuroscience research is a very active and competitive field, scientists are very careful in exposing their ongoing research projects and their data in advance. So weakness in the platform security might discourage usage.

Availability of the system. As the system is envisioned to be integrated into the scientist's daily working environment, it must not interfere with other tasks the user is performing within the system. In particular, temporary load peaks on memory and the CPUs due to image processing may slow down other applications.

Integrity of the system. As image processing methods are subject to current research, the system is explicitly laid out for integration of new methods without any need to alter the system itself. It implies that the system must be open in the sense that arbitrary code can be executed in the system. This is a high security risk, as malicious or unstable code may affect the system's integrity and stability. Code may delete or modify important system data or serve as an information leak. So security measures against these system vulnerabilities must be taken.

3. Methods.

This section explains the components of the research infrastructure and the applied security measures.

3.1. Components. The two main components in the system are the research platform and the cloud infrastructure. To keep the different concerns of the two user groups – clinical researcher and computer scientist – well separated, we decided to integrate separated components for code and job management. As a design principle, the components are loosely coupled and use standard tools and protocols wherever possible. The system encompasses different aspects of services: The underlying resources are provided as infrastructure as a service (IaaS), the deployment of new algorithms is managed by platform as a service (PaaS) and the actual

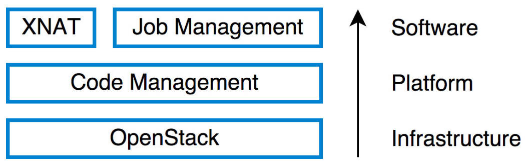


Fig. 3.1. *QMROCT components and their service categories.*

quality assessment is implemented as software as a service (SaaS). Different components of the system contribute to the different service models (cf. Fig. 3.1).

3.1.1. Infrastructure as a Service.

OpenStack. OpenStack is an Infrastructure-as-a-Service platform. It allows for on-demand creation and utilization of virtual servers, storage and network. It incorporates the management in form of multiple APIs with overall user and role management. OpenStack provides both command line tools as well as a web interface to ease the use of the REST API, that allows for managing users, projects, virtual servers, network and storage. Cloud infrastructure components dedicated for computing purposes (so called compute nodes) provide the resources (CPU and memory) to execute virtual environments that are tailored for their computation task. Load can be balanced throughout the system and therefore peaks in the required computational power can be handled. The system scales horizontally by adding additional compute nodes which will enhance the systems total memory capacity and processing power. A crucial part in this scenario, where computing resources can be created on demand and required data storage capabilities are assigned on runtime, is the network infrastructure, which is responsible for this high dynamic processing. All compute nodes are connected through a management IPv4 network, which enables communication with the control components. Everything regarding dynamic storage and computation is handled in virtual networks on top of a physical tunneling network. It connects the compute nodes with a dedicated network node. The network node creates internal bridges that will virtually route traffic through the physical network and deliver it to the virtual environment's network port that is handled entirely inside the compute node. For securing the virtual networks, Linux iptables are used. The current prototype uses OpenStack *Juno* on Ubuntu 14.04 servers. All management services are installed on the control node, except the compute services and the network service, that run on dedicated servers. To date, one control node, one network node and four compute nodes are provided. OpenStack provides two built-in mechanisms for load balancing. The Nova Scheduler handles hardware resources provided by the connected compute nodes and uses filter rules to decide which node is chosen to run a virtual server. The second mechanism is the Heat API, integrated into OpenStack since the Havana release[1]. Heat is able to run a stack of one or more virtual servers for a given application, based on a configuration file. If the load on one virtual server is too high, Heat can start a second machine to split up the load. Since OpenStack is designed for huge corporate clouds, both mechanisms assume that the infrastructure provides unlimited resources. In particular in private clouds, as employed in the presented platform, resources are often limited. A request to start a new virtual server fails and produces an error message, if there are not enough physical resources available. Virtual computation instances that can be launched by the cloud customer are referred to as *virtual servers* in this manuscript. We distinguish regular *virtual machines* from *Docker containers*, which can both be used as virtual servers in OpenStack. For testing new features, a test environment using *Devstack* is employed [38]. Devstack provides an install script for deployment of a complete OpenStack cloud infrastructure on a single physical or virtual machine. A Vagrant-based automated install script for virtual Devstack instances with Docker integration has been created for easy reproducibility and is available on GitHub [39, 40].

Docker. The employment of full virtual machines has considerable performance drawbacks in the given use case, as is shown in the results in Sect. 4.4. Successors of that approach are Docker [27] based Linux containers. They have been shown to perform better than other virtualisation concepts by Estrada and others as well as Felter and coworkers [28, 29]. Docker is a fairly new technology – the first release was March 2013 – but is based on the much older concept of operating system-level virtualisation. This special type of virtualisation is

[1]`https://wiki.openstack.org/wiki/Heat`

Table 3.1
Structure of the CMS API.

Method	**Path**	**Description**
GET	/files	Lists all files
GET	.../matlabs	Lists all Matlab files
POST	.../matlabs	Adds a new Matlab file
GET	.../file.m?compile	Compiles the specific Matlab file
DELETE	.../file.m	Deletes the specific file
GET	.../dicoms	Lists all DICOM files
GET	.../executables	Lists all executables
GET	.../binary?execute	Executes the specific binary

possible in Linux systems providing the Kernel features *control groups* and *namespaces* [30]. Both features are part of the Linux Kernel since 2008 and can be regarded as stable [31, 32]. Docker uses the term *container* for isolated environments, i.e. a control group with a namespace. A container has its own process tree, network interface (and so an own IP, routing table and iptables rules), file system and resources, such as memory, CPU and I/O. Docker adds some useful features to Linux containers: Dockerfiles holding a sequence of instructions to assemble a container image, versioning of images, a REST API, and in particular a driver for OpenStack.

Since the Havana release, OpenStack provides a hypervisor driver for Docker that is expected to be fully integrated into Kilo, the next major OpenStack version [34]. With this driver, Nova treats Docker containers like virtual machines, making it possible to use them in the same way as regular virtual machines.

3.1.2. Platform as a Service.

Code Management Service. The Code management service (CMS) is the main interface for the algorithm developer to upload, test and provide their Matlab code as stand-alone executables. The graphical user interface is implemented as a single page web application which is loosely coupled to a RESTful web service in the back end. This API offers basic file management, such as to read, delete or create files, as well as to compile and execute them. The whole functionality of the code management service is given in Table 3.1. The services are implemented as a PHP application that runs on an Apache Webserver, secured by password based authentication and HTTPS.

3.1.3. Software as a Service. The execution of the quality assessment analysis is managed by different services and components within the system.

XNAT. The main entry point for the clinical researcher is XNAT, where he or she can upload image data and start the quality assessment analysis. XNAT is implemented as a Java Enterprise web application, employing PostgreSQL as database system. The experiments presented in this paper are carried out on an XNAT 1.6.2.1 instance, using Apache Tomcat 7, Oracle JDK 1.6 and PostgreSQL 9.1 on a Debian 6.0 server instance. The data structures are designed to support typical research collaborations: researchers, projects, subjects, experiments etc. Data may be shared between different projects, and users can be assigned to the project with fine grained access rights. While especially supporting DICOM images and reports, all kinds of data types can be stored as freely definable key-value pairs (where the key is a name and the value a number), which is used in this project to implement the different quality management parameters. New parameters can be added at any time.

Besides the graphical user interface, a built-in DICOM interface and a comprehensive REST API [25] are provided. XNAT provides a pipeline engine, that can execute external applications and shell scripts. It may pass all required parameters to the application to allow for download the data, process it and upload the results back to XNAT. The pipeline engine is used within the presented infrastructure to initiate cloud-based processing. However, due to certain limitations of the built-in pipeline engine, in particular sequential processing of pipelines, the actual monitoring and scheduling of the cloud job is taken over by a dedicated cloud-aware job management service.

Image Upload. To ease the image upload for the clinical researcher, a dedicated image gateway is installed within the clinics. It receives images from the OCT scanner, blurs some patient related metadata of the OCT images to minimize the reidentification potential, compresses the images and sends them via the REST interface to the external XNAT instance. Once the images are received by XNAT, it extracts the data and stores them in the prearchive. Two further steps are automatically executed: (1) The images must be marked as "ready" for archiving, before they are moved from the prearchive into the archive; (2) Once arrived at the archive, the images can be processed.

Job Management Service. The execution of the cloud-based image analysis methods is carried out by the job management service (JMS). It launches virtual servers depending on hardware availability, initiate job execution and terminate the servers when the job is finished. A particular feature is resource-aware scheduling. Because the built-in OpenStack tools do not provide functionality to resolve issues with limited resources, a lightweight Python-based cloud scheduler has been developed. It provides a REST API to enqueue incoming jobs and processes them whenever there are free resources. The scheduler is also able to limit the amount of virtual servers running simultaneously and to reuse them several times for different jobs if needed.

QMROCT VM image and Docker container. The quality assessment software itself is delivered within specifically configured virtual servers. In particular, the Matlab Compiler Runtime 2013b and the currently provided Matlab executables themselves are installed. We would like to emphasise, that the usage of Matlab standalone applications, i.e. compiled Matlab code, allows to deploy the code multiple times even in public clouds without licensing issues, as the free-of-charge Matlab Compiler Runtime is used. Furthermore, SSH is used to access the server. The QMROCT VM image is based on the official Ubuntu 14.04 cloud image [36]. It incorporates the Cloud-Init package [37], providing a lot of functionality to OpenStack, like resizing the image partition and inserting a public key for SSH authentication. On the downside, this results in a slower boot process for the virtual machine.

The QMROCT Docker container is based on a standard Ubuntu 12.04 image. It also contains the Matlab Compiler Runtime and executables, but not the Cloud-Init package, because a Docker container does not need features like repartitioning. Therefore the Docker container only runs an SSH daemon. A public key for authentication cannot be copied to the instance on start-up without Cloud-Init. To overcome this issue Docker is configured to initialize a key, when building the container image.

3.2. Security measures. The security assessment of the system follows the so-called *IT-Grundschutz* (IT basic protection) methodology of the Federal Office for Information Security (BSI Standard 100-2) [41], which interprets the requirements of the ISO Standards of the ISO 2700x family [42]. The basic idea of IT-Grundschutz is, that most IT systems can be modelled as a combination of well-known building blocks like servers, clients, network components and applications. For each of these building blocks, there exist typical security risks and organizational and technical security recommendations, that are collected in the *IT-Grundschutz* catalogues. The first step in such a security assessment is the analysis and documentation of the addressed business processes, encompassing the involved data, human actors and applications. The information processed is then categorized in different *protection levels*, depending on the possible harm due to loss of confidentiality, integrity or availability of the respective information. Now the related IT components are identified and are mapped on the existing building blocks. A subsequent basic security check compares the target and current state of the system regarding the security recommendations. A work plan is then defined to efficiently implement higher levels of information security.

4. Results. The resulting quality assessment platform provides a separated environment for clinical researchers and code developers. It is analysed in terms of functionality, usability, security and performance.

4.1. Functionality. Both the OCT quality assessment initialized by the clinical researcher as well as the integration of new image processing methods by the computer scientist were successfully implemented. An overview of the architecture is given in Figure 4.1. The so-called QMROCT Gateway is the central component to link XNAT to OpenStack through the JMS service. It also provides the code developer interface. The system's functionality is described along the two user scenarios, the quality assessment and the code upload.

4.1.1. Quality Assessment. The process of quality assessment on new images is shown in Figure 4.2. The clinical researcher uploads new images to XNAT. XNAT can be configured to trigger the quality assessment pipeline automatically, whenever new images have been uploaded, or to start the pipeline only on user interaction. XNAT's built-in pipeline system calls a small program, which sends the corresponding XNAT session

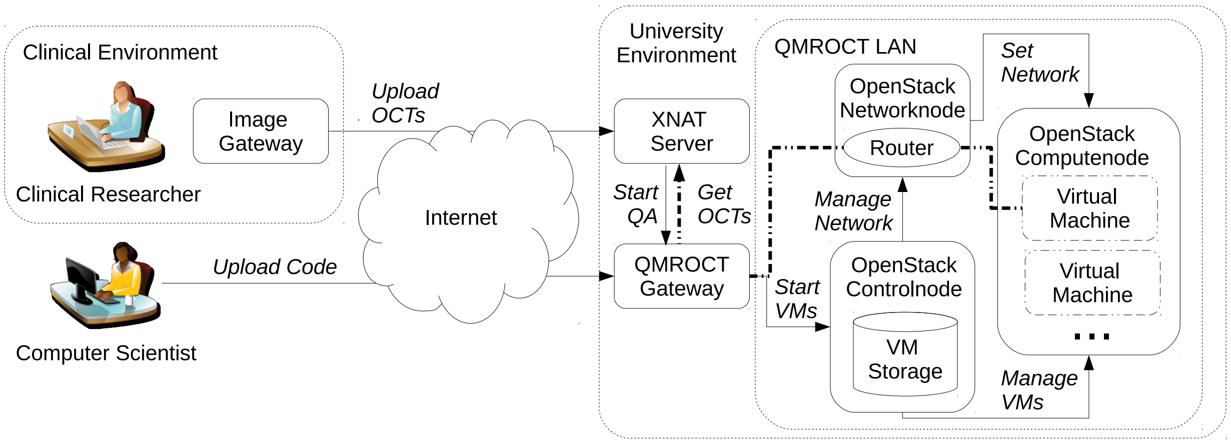


FIG. 4.1. *Architecture of the QMROCT infrastructure.*

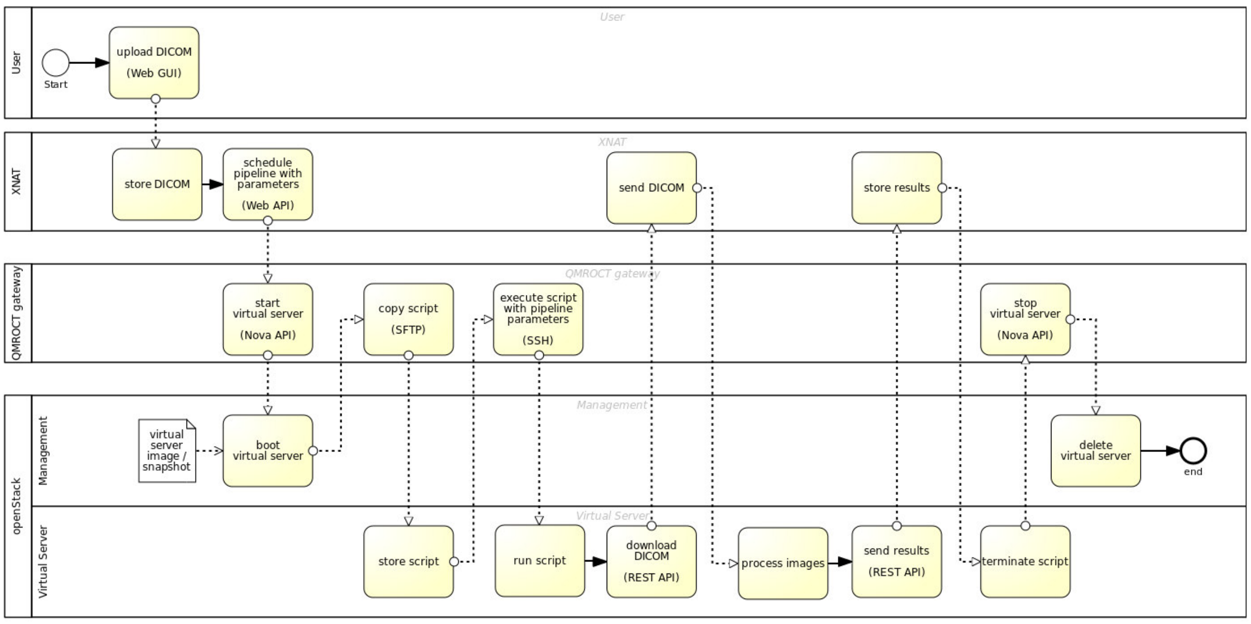


FIG. 4.2. *Process of the clinical researcher scenario: Quality assessment of new OCT images.*

parameters as JSON object to the JMS. The JMS handles the further job processing with the resources provided by OpenStack. After launching a virtual server, the QMROCT shell script is copied to the server and is executed with the given pipeline parameters via the SSH interface. The QMROCT shell script then connects to the XNAT server specified in the pipeline parameters and downloads the OCT images. These images are processed with the MATLAB executables present on the virtual server; and the results encompassing the numeric *Key Quality Indicators* and a DICOM result image are uploaded to the corresponding XNAT image session. When the processing is finished, the script terminates and the JMS shuts down the virtual server.

4.1.2. Code Upload. The process of uploading and testing code is shown in Figure 4.3. The software developer uploads new Matlab scripts to the CMS using the web interface, that is shown in Figure 4.4. Single

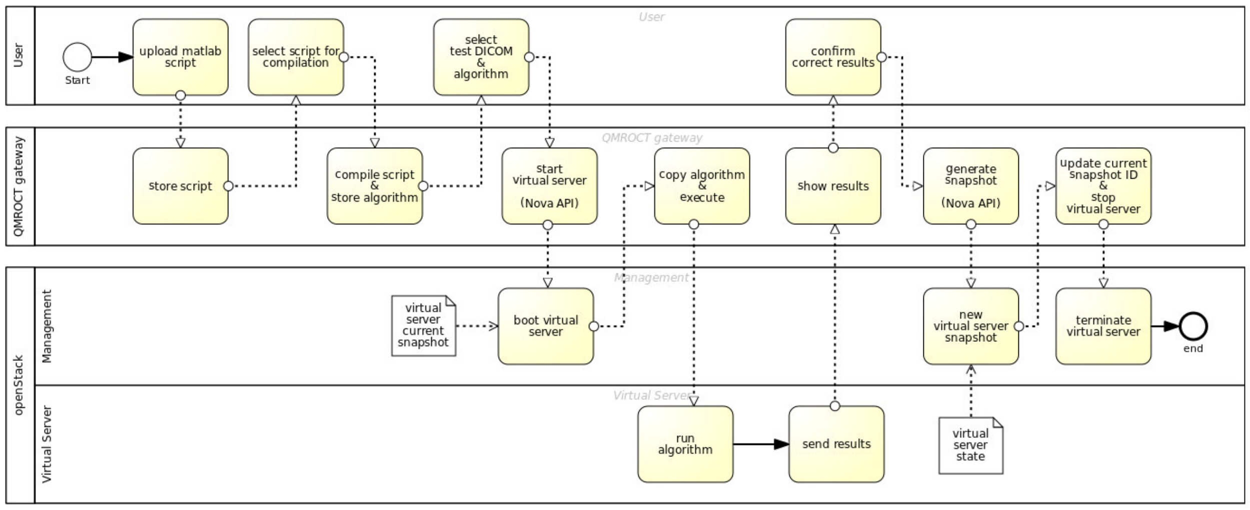

User
QMROCT gateway
openStack
Management
Virtual Server
Start
upload matlab script
select script for compilation
select test DICOM & algorithm
confirm correct results
store script
compile script & store algorithm
start virtual server (Nova API)
copy algorithm & execute
show results
generate snapshot (Nova API)
update current snapshot ID & stop virtual server
virtual server current snapshot
boot virtual server
new virtual server snapshot
terminate virtual server
end
run algorithm
send results
virtual server state

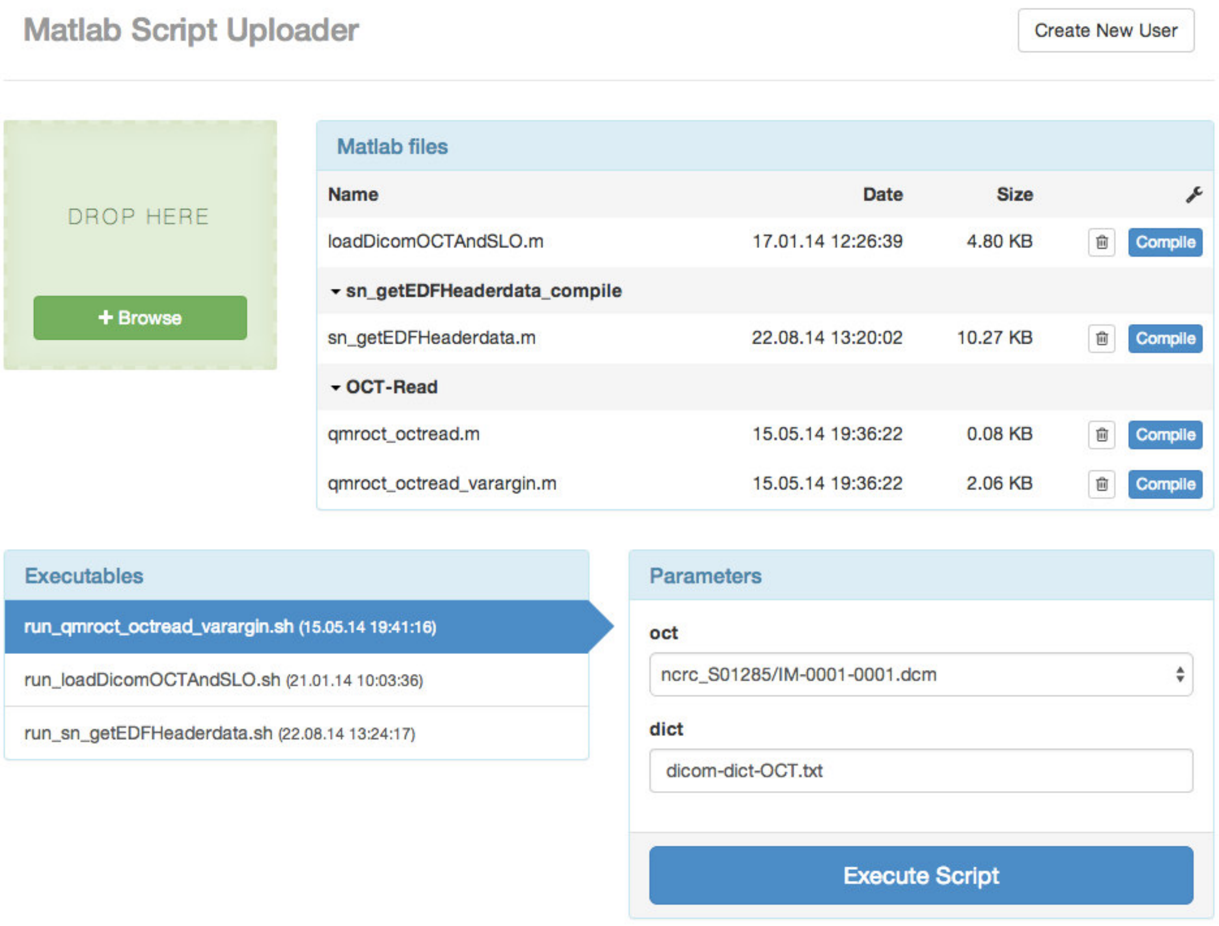
Fig. 4.3. *Process of the algorithm developer scenario: Compilation and testing of new Matlab code.*

scripts or zip-archived folders containing script collections are supported. These scripts are then stored in a local Git repository for version control. He or she may then select a script for compilation. When successfully compiled, the executables are added to the executable archive on the gateway, and a Matlab-generated run script for the new executable is shown in the list of available methods. Every algorithm needs to be tested before it can be used for quality assessment: The user selects a run script, chooses – depending on the method – one or more test images and further parameters and starts the execution manually. Then the new method is executed on the server and results or possible errors are displayed in the user interface. Key quality indicators are shown in the user-interface, the DICOM result image can be downloaded following the given link. When no errors occurred and the results are identical to those in the local environment, the developer may confirm the results, and the new method is integrated into the QMROCT virtual server.

4.2. Usability. The user interaction within the QMROCT platform is designed to be very simple. Within XNAT, the cloud-based processing is completely transparent and appears to be a standard XNAT pipeline. The clinical researcher accesses the cloud services and resources within his or her daily working environment. The only step that users need to do is selecting the image data to be processed and then submit the job, which will automatically start in the background. No use of command-line based tools, no manual installation and configuration of software clients or even complicated input of technical configuration data are needed by the clinical researcher.

With the CMS, several scripts can be uploaded simultaneously by simply dragging and dropping them onto the upload area. Timestamps are added to the scripts and executables to give the user an overview of the status of the uploaded code. The only convention, the developer has to follow, is to add a specified signature line to the Matlab script that includes the name of the function and – based on a mutually agreed list – the type of every parameter. The signature line is parsed by the upload tool to offer the user – after he or she selected an executable – a list of all available parameters, pre-filled with data based on an educated guess, e.g. a list of test DICOM images or the default DICOM dictionary file. The results of the code execution will be displayed directly in the web interface and can be compared to the local reference data. If a developer deems an algorithm mature, he or she can approve it for production usage. The whole process is completely independent from XNAT and does not interfere with the work of the clinical researchers.

Regarding the extensibility of the platform for the system developers, the strict service oriented architecture (SOA) offers an easy-to-use solution to add new services into the system, for example further user interfaces or additional cloud services.


Matlab Script Uploader

Create New User

DROP HERE

+ Browse

Matlab files

Name	Date	Size
loadDicomOCTAndSLO.m	17.01.14 12:26:39	4.80 KB
▾ sn_getEDFHeaderdata_compile		
sn_getEDFHeaderdata.m	22.08.14 13:20:02	10.27 KB
▾ OCT-Read		
qmroct_octread.m	15.05.14 19:36:22	0.08 KB
qmroct_octread_varargin.m	15.05.14 19:36:22	2.06 KB

Executables

run_qmroct_octread_varargin.sh (15.05.14 19:41:16)
run_loadDicomOCTAndSLO.sh (21.01.14 10:03:36)
run_sn_getEDFHeaderdata.sh (22.08.14 13:24:17)

Parameters

oct: ncrc_S01285/IM-0001-0001.dcm

dict: dicom-dict-OCT.txt

FIG. 4.4. *CMS Web Interface to upload and test new Matlab code.*

4.3. Security. Following the IT-Grundschutz methodology, the processes and involved systems are identified as described in this manuscript (cf. Sect. 3.2). The different user roles identified within this system are shown in Table 4.1. The processed information includes possible metadata, code, and user activities. The confidentiality level is medium, rather than high, as the image data are pseudonymised. The same holds for the availability level and the integrity level. To mitigate the consequences of a system breakdown, all image data processed within the system are also stored locally in the researcher's PACS and the source code is also available in the developer's local environment. So typical security measures are supposed to be sufficient for the system.

4.3.1. Confidentiality. As mentioned before, all communication through the internet is encrypted via TLS, encompassing user and machine access. Within the OpenStack infrastructure, no component is directly connected to the internet. In particular, the virtual servers run in their private network and may connect to the internet via a virtual router on the network node and a subsequent router on the QMROCT gateway. The so-called "floating IP" address, which is assigned to a virtual server, can only be accessed from the QMROCT gateway. An OpenStack security group restricts this access to allow only SSH. Any virtual machine or container can initialize an internet communication with XNAT, but they cannot be called from outside. Due to usability reasons, PKI-based user authentication and authorisation is not employed. As cloud infrastructures are designed for multi-tenant use rather than collaborative resource sharing, the separation between different users is much stronger than in typical academic Grid infrastructures, lowering the risk of information leaks. All code is executed within the virtual environments which are always instantiated for one specific task and terminated afterwards, so no information about former activities is stored there. From a security perspective, the Docker containers behave similar to virtual machines, as they shield different instances from each other. A container can not manipulate another, because it only knows about it if declared explicitly a priori, i.e. by linking containers. As of now there is no known way for a process to break out of a Docker container, but a potential risk is posed by the Docker daemon as it has to run with root privileges. There is work in progress to change namespaces to run a container as an unprivileged user [33], lowering the mentioned risk. On XNAT and OpenStack, the built-in password-based authentication and role-based access control methods are used (cf. Table 4.1).

4.3.2. Integrity. The risk of malicious manipulation of data can be regarded as comparatively low. As data transfer is encrypted, man-in-the-middle attacks are difficult, and all user activity on the different components – the XNAT and the CMS/JMS services – are logged. A higher risk might be the unintended assignment of image data and results to a wrong subject. As the virtual servers processing the image data only know the parameters of the assigned pipeline, it has no information about other resources from XNAT. Therefore it is very unlikely, that QM results are stored along the wrong subject or image data. In case a mismatch would occur, it can be identified by the overview image that is also provided as result. The code, the results and the virtual machine snapshots are under version control, so in case of undesired modification, a former version can be reconstructed.

4.3.3. Availability. Main risks for availability are network interruptions and server failures. The production network infrastructures of the participating institutions are employed, leading to very few network problems. To avoid availability issues because of the code execution – may it be due to high CPU load or due to malicious or unstable code – all processing steps are transferred to the cloud environment. In addition, the use of virtual servers increases the stability of applications, since the system's configuration is largely independent from the host system of the cloud provider. In case of a successful attack on a virtual server, it is hardly possible to get access to the host system, so there is very low risk that the platform and application availability is affected. To account for possible data-loss, the XNAT platform is integrated into the institution's backup system, while the code stored on the QMROCT gateway is duplicated in the QMROCT VM and Docker images.

4.4. Performance. Performance tests have been carried along the typical clinical researcher's use-case of the infrastructure: At the end of an examination day, the images are uploaded to the infrastructure and are processed. The tests are performed with 12 OCT image sets. Each image set consists of an optical overview image and a tomographic scan of the retina. The overview images have always a size of 580 kB, the tomographies' sizes range from 14 MB to 27 MB; adding up to 294 MB for the full dataset.

Currently, three quality assessment methods are implemented within the infrastructure, encompassing poor signal strength and centration of the scan. We would like to emphasise that the algorithms are not developed by the authors but by the project partners (cf. 6). Method 1 and method 3 each provide a result image visualising the regarding quality indicators, method 1 and method 2 provide key image indicators in form of key-value pairs. The result images' sizes range from 700 KB to 2 MB, the key quality indicators are written in a text file, encompassing few bytes. Each full processing from pseudonymisation of the image to quality assessment result storage back to XNAT has been performed ten times. As the quality assessment pipeline could not be started remotely via the REST API, the tests are subdivided into the upload process and the quality assessment process.

Image Upload Process. The image upload process encompasses five different steps (cf. 3.1.3). The duration of the different steps are given in Figure 4.5.

The whole image upload process requires less than 2 minutes in all runs, with an average runtime of (94 $\pm$ 6) seconds. The main contributions are the data compression and transfer. Interestingly, decompression on the XNAT server is much faster. The data transfer took in average (65 $\pm$ 6) seconds, giving a data transfer rate about 4.33 MB/s. Compared to other modalities, OCT images are small, so data transfer from the clinics to the quality assessment infrastructure will not be a severe performance issue. But with an estimated data transfer of about 350 GB per day, the current setup does not support medical imaging methods that generate data in a magnitude of terabytes.

TABLE 4.1
Defined user roles with their permission in the QMROCT infrastructure.

Role	**Rights**	**Data Access**
	End User	
Clinical Researcher	Can upload images, execute analysis jobs and search images via XNAT web portal	Images inside her or his project and analysis results
Computer Scientist	Can upload new image analysis algorithms via gateway server	Source code and compiled code on the QMROCT gateway
	Administration	
XNAT Administrator	Can change configuration of XNAT and activate/deactivate XNAT and user access to fix bugs or improve system safety	Log files of the application, configuration data, data of portal users, no access to application data
Server Administrator	Can update and reboot the server and services	Information about users, processes and images stored on the file system
Application Administrator	Can change application configuration, activate/deactivate components of the application	Log files of the application, configuration data, no user data
Cloud Administrator	Can change cloud components and user management and deactivate/start cloud services to fix bugs and improve system safety	Log files of component configuration data, data of user management, no access to application data and configuration data (image data) of cloud services
	Development	
JMS/CMS Developer	Can change code and job management services and user interface to fix bugs or improve performance of the application	Data necessary for the execution of application functionality, possibly log files, access to user data or image data stored within the service
XNAT Developer	Can change the application code to fix bugs or improve performance of the application	Log files of the application, configuration files, possibly data contents related to application performance, no personal data access
Cloud Service Developer	Can change services and update libraries to fix bugs or improve performance of the services	Log files of the service, configuration files, possibly data contents related to application performance, no personal data access, boot access to hardware

Quality Assessment Process. The quality assessment process is realised using two OpenStack compute nodes with Intel(R) Core2 Duo (3 GHz) CPUs. One compute node is equipped with 8 GB RAM, the other one with 4 GB RAM. The VMs described in paragraph 3.1.3 are instantiated with 2 GB of RAM. The JMS is configured to allow a maximum of four VMs at a time. We would like to emphasise that overloading the compute nodes is possible, but the current bottleneck is the provision of floating IPs, which may fail if more than four IPs are requested at the same time. The overall runtime of the quality assessment process is shown in Figure 4.6.

The whole quality assessment process requires in all runs less than 16 minutes, with an average runtime

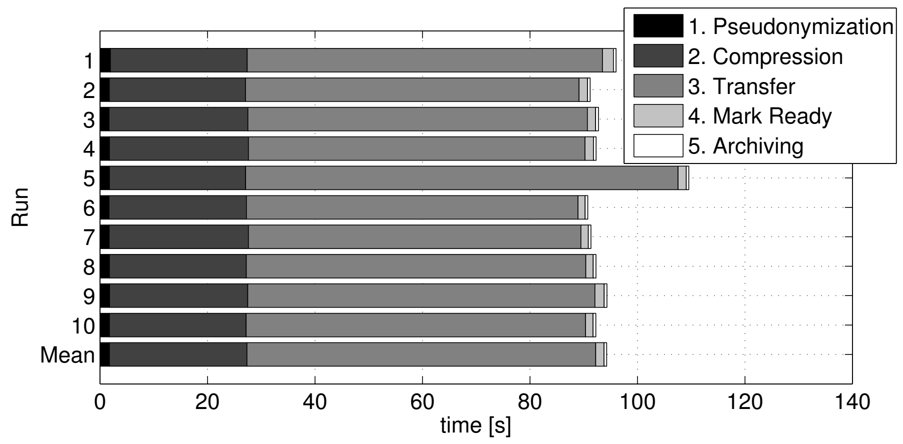


FIG. 4.5. *Runtime of the image upload process of 12 image sets with a total size of 294 MB.*

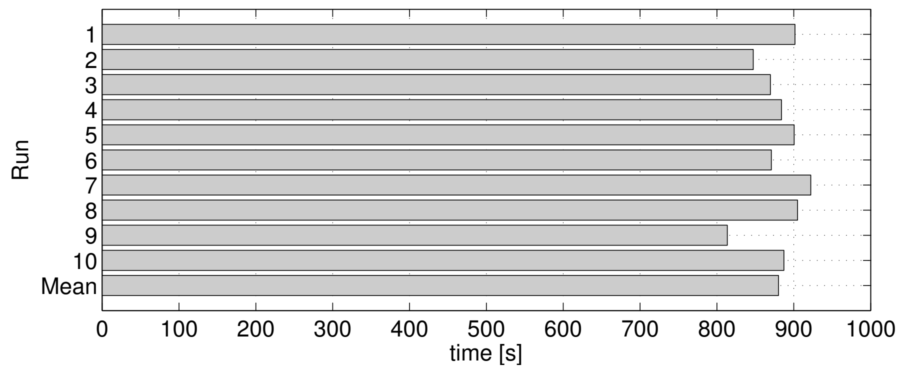


FIG. 4.6. *Runtime of the quality assessment process of 12 image sets with a total size of 294 MB and three image processing methods.*

of 14.6 minutes, or (880 ± 30) seconds. So the whole image pipeline from pseudonymisation to result storage requires less than 20 minutes for all 12 images.

To get a closer look into the execution times of the different processing steps, Figure 4.7 shows the runtimes of all individual quality assessments.

The quality assessment for a single image set requires always less than 13 minutes, in average about 5 minutes or ((280 ± 46) s). Compared to the average runtime of the complete set of 12 image sets, the distributed system provides a speed-up of factor 3.1 compared to sequential processing. A maximum limit of 4 is given by the current settings of the scheduler. The processing itself with an average of (190 ± 34) s takes the main part of the overall runtime, followed by the instantiation of the VM ((76 ± 18) s). The data transfer within the system requires an average of (12 ± 5) s for each individual image set. Figure 4.8 shows the mean runtimes for the individual image sets.

The differences in runtime might on one hand be caused by image properties, like the image size, which would effect the data transfer and possibly the execution times of the quality assessment methods; or intrinsic image features like the signal-to-noise-ratio, which might affect the execution times of one or more quality assessment methods. Table 4.2 shows the different image sizes of the processed images.

As can be seen by comparison with Figure 4.8, there is no obvious correlation between the image size and the data transfer time. On the other hand the runtime might be influenced by the availability of cloud resources. In particular, we see an increasing prolongation of the runtime for images 2, 3, and 4. We assume, that this is caused by the simultaneous start of the first four virtual machines. Later, due to differences in the execution times of the quality assessments, the resource allocation is desynchronized for the four concurrent processes, so

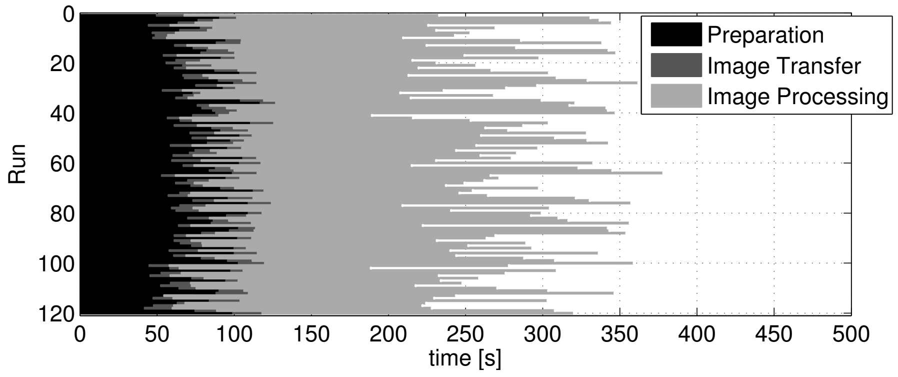


FIG. 4.7. *Runtimes of the individual quality assessments for single image sets.*

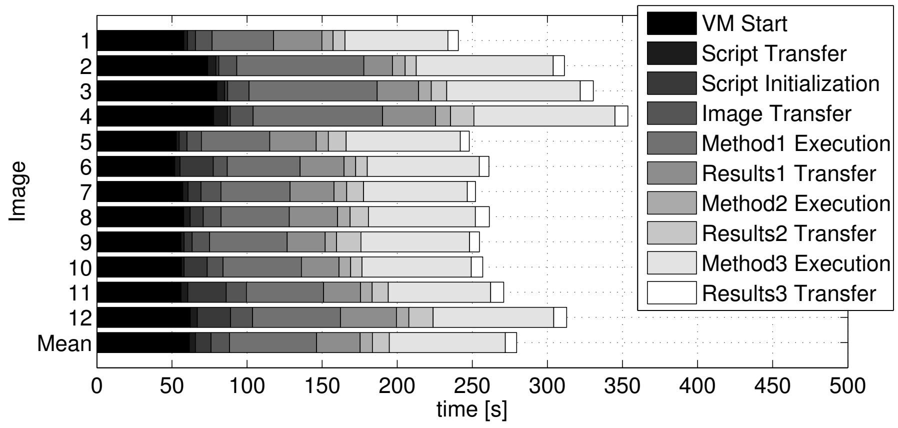


FIG. 4.8. *Average runtimes for the quality assessment of the different image sets.*

VM instantiation and data transfer is better spread in time.

Docker containers. While performance optimisation of the algorithms is up to the algorithm developers, on the system side the starting of the virtual machines takes the main part of the management overhead. This performance could be significantly increased by using Docker containers. To compare the performance of Docker containers and virtual machines, additional tests have been realised. These performance tests have been carried out within a single virtual Ubuntu 14.04 Devstack machine equipped with 8 GB RAM and four virtual CPUs assigned. The OpenStack version used with Devstack is *Icehouse*, the last release before *Juno*. For each test, ten strongly simplified data processing pipelines are scheduled at once. They are then processed in sequential or in parallel mode. Parallel mode schedules the pipelines up to a fixed number of two simultaneously running jobs. As the focus of this test was on the mangement overhead, no data transfer nor image processing was included, but a test script writing a string to a file was employed. For each pipeline a virtual server is booted and a small test script, which only writes a string to a file, is copied to this server and then executed. For every test the total time for processing all ten pipelines and the time each virtual server is running have been measured. The virtual server runtime is measured from booting the server until the test pipeline script has been executed and terminates. Every test is carried out five times. The results are given in Figures 4.9 and 4.10. We would like to emphasise, that the results cannot be compared directly with the performance tests above. As the Docker support in OpenStack is still in active development, we performed these tests in a test environment.

As can be seen in Figure 4.9, the system scales well, but not linearly. While the average total time for a sequential execution of ten runs with regular virtual machines is about 47 minutes, parallel execution reduces the execution time to 27 minutes (about 60%). Using Docker, the average total time is close to 5 minutes

TABLE 4.2
Image sizes of the tomographic images.

Image	1	2	3	4	5	6	7	8	9	10	11	12
Size [MB]	27	23	23	23	27	27	23	14	23	27	23	23

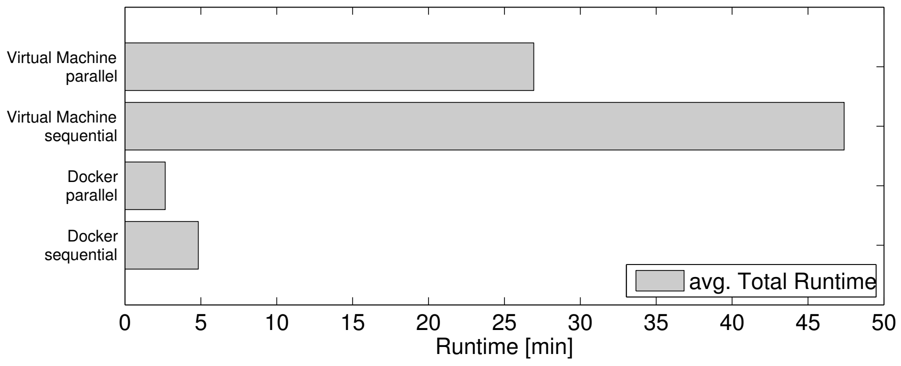


FIG. 4.9. *Performance tests comparing the average total time of executing 10 pipelines in virtual machines and Docker containers, each in sequential and in parallel mode.*

in sequential and about 2.6 minutes in parallel mode, requiring about 10% of the time with respect to the respective tests using virtual machines.

The average runtime for each virtual server is shown in Figure 4.10. For both, virtual machine and Docker container, the runtime is slightly longer in parallel mode compared to sequential execution. This measurement correlates with the results from Figure 4.9, where the parallel execution requires slightly more than 50% of the sequential execution. It can be explained by the inherent resource sharing: The parallel execution uses more computing capacities, which results in a slightly slower execution of an individual run.

5. Conclusion and Outlook. The envisioned quality assessment platform for OCT images has been set up successfully. The system is currently used by the clinical researchers from the local university clinics participating in the QMROCT project. In order to increase functionality and usability and to minimize specific security problems of the neuroimage processing, we have integrated existing and newly developed platforms and components to the QMROCT environment. Both the image processing pipeline as well as the test execution of new code requires only few interaction steps with the user: The quality assessment requires the data and pipeline selection, the code testing requires code and data selection and optional parameter settings, the results are automatically transferred to the respective target component and user interface. The QMROCT platform has been analysed with respect to IT security and many recommendations from the *IT-Grundschutz* catalogue have been implemented. However, as the specific applications cannot be modelled with standard building blocks, and as new security vulnerabilities in underlying services and tools are reported regularly, system security is a continuous process. Anyhow, the most vulnerable step regarding patient data protection, the automatic upload of image data from the clinical environment is not yet fully employed and is ongoing work.

According to the performance results, the quality assessment of a typical examination day can be processed within 20 minutes. Currently, three quality assessment methods are available. However, as new methods are continuously integrated, the processing time will eventually increase. On the infrastructure, the instantiation of the virtual machines takes the largest share of the management overhead. The employment of Docker containers significantly reduces the management overhead to about 10% with respect to virtual machines. Reuse of virtual servers in large-scale image processing is envisioned, but is to be examined regarding confidentiality, as it would violate the disposable-virtual-server approach, that currently wipes any image-specific information from the cloud infrastructure directly after job execution. The integration of Docker into the production environment is currently realised, so full production-like performance tests need to prove these findings. Furthermore, the employment of about 40 computers from student's labs as OpenStack compute nodes is envisioned, allowing

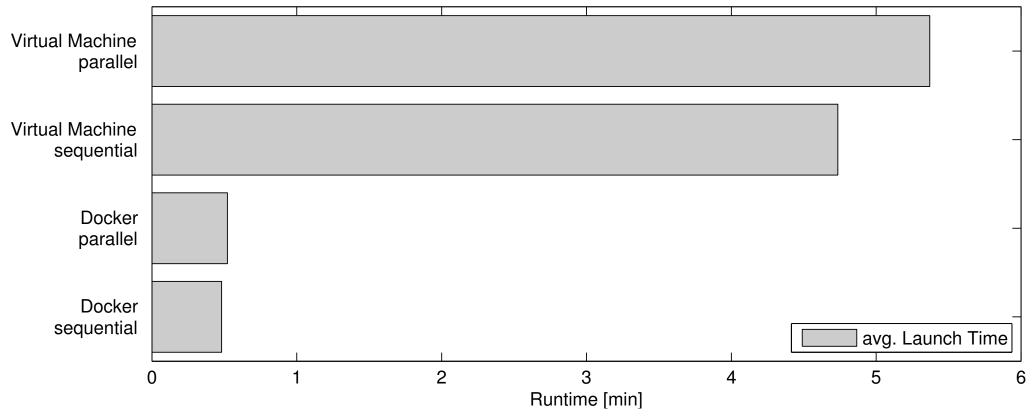


Fig. 4.10. *Average launch time of each virtual server during the performance tests.*

for faster processing of large data sets. While the JMS is capable of handling limited resources and common errors, a descent workflow manager monitoring the jobs to give detailed information about process status, cloud related errors or wrong usage of resources is to be implemented. Many workflow management tools like Pegasus [43], MOTEUR [44], Taverna [45] or WS-PGRADE were developed for grid infrastructures and are continuously supporting more and more cloud infrastructures. OpenStack itself recently published a new workflow management service called Mistral [46], that we will evaluate. We would like to emphasise that the QMROCT platform – except the XNAT extensions for the quality measure data – is not research-domain specific. In particular, other science gateways might be connected to the JMS and CMS. Currently, further neuroimage and biosignal processing scenarios are implemented using the same infrastructure.

6. Acknowledgements. We would like to thank all participants of the QMROCT project, in particular Frank Haußer and Inge Beckers from Beuth University of Applied Sciences Berlin for leading the quality assessment algorithm development and Alexander Brandt, Timm Oberwahrenbrock, Hannah Zimmermann and Ella Kadas from the Neuroscience Research Center of Charité - Universitätsmedizin Berlin for providing the OCT images and medical expertise.

7. Source Code. The developed components are open source projects and available on the QMROCT GitHub page:

1. Pipeline-Scheduler [35] (GNU General Public License)
2. Vagrant Box including Devstack with Docker integration [40] (MIT License)
3. Docker Container including Matlab and an SSH daemon [47] (MIT License)

REFERENCES

[1] A. Petzold et al., *Optical coherence tomography in multiple sclerosis: a systematic review and meta-analysis*, Lancet Neurology, Sep. 9, 2010. Accessed: Sep. 11, 2014. Available: http://www.thelancet.com/journals/laneur/article/PIIS1474-4422%2810%2970168-X/abstract

[2] A. U. Brandt et al., *Primary retinal pathology in multiple sclerosis as detected by optical coherence tomography*, Brain, Feb., 2013. Accessed: Sep. 11, 2014. Available: http://brain.oxfordjournals.org/content/134/11/e193.extract

[3] B. Fischl, *FreeSurfer*, Jan. 1, 2012. Accessed: Sep. 10, 2014. Available: http://www.sciencedirect.com/science/article/pii/S1053811912000389

[4] *FreeSurfer Beginners Guide*. Accessed: Sep. 10, 2014. Available: http://freesurfer.net/fswiki/FreeSurferBeginnersGuide

[5] P. Tewarie et al., *The OSCAR-IB consensus criteria for retinal OCT quality assessment*, Apr. 19, 2012. Accessed: Sep. 11, 2014. Available: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0034823

[6] *DICOM*. Accessed: Feb. 9, 2014. Available: http://medical.nema.org/standard.html

[7] D. S. Marcus et al., *The extensible neuroimaging archive toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data*, Neuroinformatics, 2007, Springer. Accessed: Sep. 11, 2014. Available: http://link.springer.com/article/10.1385/NI%3A5%3A1%3A11

[8] J.-B. Poline et al., *Data sharing in neuroimaging research*, Apr. 5, 2012, Frontiers in Neuroinformatics. Accessed: Sep. 10, 2014. Available: http://journal.frontiersin.org/Journal/10.3389/fninf.2012.00009/abstract

[9] S. Seifert et al., *Semantic annotation of medical images*, Mar. 11, 2010, Medical Imaging 2010: Advanced PACS-based Imaging Informatics and Therapeutic Applications. Accessed: Sep. 10, 2014. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=748216

[10] Microsoft *Azure*. Accessed: Sep. 10, 2014. Available: http://azure.microsoft.com

[11] Amazon, *AWS, Amazon elastic compute cloud (EC2)*. Accessed: Sep. 10, 2014. Available: http://aws.amazon.com/

[12] Google, *Google Cloud Platform*. Accessed: Sep. 10, 2014. Available: https://cloud.google.com/

[13] OpenStack Foundation, *OpenStack*. Accessed: Sep. 10, 2014. Available: http://www.openstack.org/

[14] P. Mell and T. Grance, *The NIST definition of cloud computing*. Accessed: Sep. 10, 2014. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[15] J. Wu et al., *The Charité Grid Portal: User-friendly and Secure Access to Grid-based Resources and Services*, Oct. 12, 2012, Journal of Grid Computing, Springer. Accessed: Sep. 10, 2014. Available: http://link.springer.com/article/10.1007%2Fs10723-012-9234-3

[16] P. Kacsuk, *P-GRADE portal family for Grid infrastructures*, Mar. 10, 2011, Concurrency and Computation: Practice and Experience. Accessed: Sep. 10, 2014. Available: http://onlinelibrary.wiley.com/doi/10.1002/cpe.1654/abstract

[17] I. Foster et al., *A security architecture for computational grids*, 1998, CCS '98 Proceedings of the 5th ACM conference on Computer and communications security, ACM Press. Accessed: Sep. 10, 2014. Available: http://portal.acm.org/citation.cfm?doid=288090.288111

[18] I. Dinov et al., *Neuroimaging Study Designs, Computational Analyses and Data Provenance Using the LONI Pipeline*, September 28, 2010. Accessed: Sep. 10, 2014. Available: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0013070

[19] R. Siewert et al., *Web-based Interactive Visualization in a Grid-enabled Neuroimaging Application Using HTML5*, 2012, IOS Press. Accessed: Sep. 11, 2014. Available: http://ebooks.iospress.nl/publication/21428

[20] C. Fiehe et al., *Building a Medical Research Cloud in the EASI-CLOUDS Project*, Jun., 2014, Science Gateways (IWSG) 6th International Workshop, IEEE. Accessed: Sep. 10, 2014. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6882066&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6881322%2F6882053%2F06882066.pdf%3Farnumber%3D6882066

[21] T. Bednarz et al., *Cloud-based image analysis and processing toolbox for biomedical application*, 2012. Accessed: Sep. 10, 2014. Available: http://www.ci.uchicago.edu/escience2012/pdf/escience2012_submission_189.pdf

[22] R. Alonso-Calvo et al., *Cloud computing service for managing large medical image data-sets using balanced collaborative agents*, 2011, Advances in Intelligent and Soft Computing, Springer Berlin Heidelberg. Accessed: Sep. 10, 2014. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-19875-5_34

[23] B. Da Mota et al., *Machine learning patterns for neuroimaging-genetic studies in the cloud*, Apr 8, 2014, Frontiers in Neuroinformatics. Accessed: Sep. 11, 2014. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3986524/

[24] J.M. Salinas et al., *R & D Cloud CEIB: Management System and Knowledge Extraction for Bioimaging in the Cloud*, 2012, Advances in Intelligent and Soft Computing, Springer. Accessed: Sep. 11, 2014. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-28765-7_39

[25] L. Richardson et al., *RESTful web services*, 2007, O'Reilly.

[26] OpenStack Foundation, *OpenStack Havanna*. Accessed: Sep. 11, 2014. Available: https://wiki.openstack.org/wiki/Heat

[27] Docker, Inc. Accessed: Sep. 12, 2014. Available: https://www.docker.com/

[28] Z.J. Estrada et al., *A Performance Evaluation of Sequence Alignment Software in Virtualized Environments*, May, 2014, 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Accessed: Sep. 11, 2014. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6846525&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6846525

[29] W. Felter et al., *An Updated Performance Comparison of Virtual Machines and Linux Containers*, July 21, 2014, IBM Research Report. Accessed: Sep 10, 2014. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

[30] J. Petazzoni, *Lightweight Virtualization with Linux Containers*, June 7, 2013, The 5th China Cloud Computing Conference. Accessed: Sep. 10, 2014. Available: http://www.ciecloud.org/2013/subject/07-track06-Jerome%20Petazzoni.pdf

[31] *Linux Kernel 2.6.24*. Accessed: Sep. 12, 2014. Available: http://kernelnewbies.org/Linux_2_6_24

[32] *Linux Kernel 2.6.26*. Accessed: Sep. 12, 2014. Available: http://kernelnewbies.org/Linux_2_6_26

[33] *LXC 1.0.5*. Accessed: Sep. 12, 2014. Available: https://linuxcontainers.org/news/

[34] OpenStack Foundation, *Docker, OpenStack Wiki*. Accessed: Aug. 30, 2014. Available: https://wiki.openstack.org/w/index.php?title=Docker&oldid=61664

[35] QMROCT, *Pipeline Scheduler*. Accessed: Sep. 10, 2014. Available: https://github.com/QMROCT/pipeline-scheduler

[36] Ubuntu Documentation Team, *Ubuntu Enterprise Cloud*. Accessed: Sep. 10, 2014. Available: http://help.ubuntu.com/community/UEC/Images

[37] OpenStack Foundation, *OpenStack Linux image requirements*. Accessed: Sep. 10, 2014. Available: http://docs.openstack.org/image-guide/content/ch_openstack_images.html

[38] OpenStack Foundation, *Devstack*. Accessed: Sep. 10, 2014. Available: http://devstack.org/

[39] HashiCorp, *Vagrant*. Accessed: Sep. 10, 2014. Available: http://www.vagrantup.com/

[40] QMROCT, *Vagrant based Devstack with Docker*. Accessed: Aug. 30, 2014. Available: https://github.com/QMROCT/vagrant-devstack-docker

[41] Federal Office for Information Security, *BSI-Standards*. Accessed: Sep. 10, 2014. Available: https://www.bsi.bund.

de/EN/Publications/BSIStandards/BSIStandards_node.html

[42] *ISO/IEC 27001:2013, Information technology - Security techniques - Information security management systems - Requirements.* Accessed: Sep. 10, 2014. Available: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54534

[43] K. Lee et al., *Adaptive workflow processing and execution in pegasus*, Nov., 2009, Concurrency and Computation: Practice and Experience. Accessed: Sep. 10, 2014. Available: http://onlinelibrary.wiley.com/doi/10.1002/cpe.1446/abstract

[44] T. Glatard et al., *Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR*, Aug., 2008, International Journal of High Performance Computing Applications. Accessed: Sep. 10, 2014. Available: http://hpc.sagepub.com/content/22/3/347

[45] K. Wolstencroft et al., *The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud*, May 2, 2013, Nucleic Acids Research. Accessed: Sep. 10, 2014. Available: http://nar.oxfordjournals.org/content/41/W1/W557

[46] OpenStack Foundation. *Mistral*, Accessed: Sep. 10, 2014. Available: https://wiki.openstack.org/wiki/Mistral

[47] QMROCT, *Docker container with Matlab and SSH*, Accessed: Sep. 10, 2014. Available: https://github.com/QMROCT/matlab-docker

Edited by: Jesus Carretero
Received: September 15, 2014
Accepted: January 21, 2015