Róbert Lovas

Application of Cloud Computing Based Platforms for Environmental and Societal Security

Róbert Lovas PhD, Senior Research Fellow and Deputy Head of the Laboratory of Parallel and Distributed Systems, Institute for Computer Science and Control of the Hungarian Academy of Sciences; Associate Professor of Óbuda University

Abstract

Environmental and societal security are two important areas in the generic topic of security issues. Numerous IT research and development achievements can provide efficient support and toolset for security experts to prevent, detect and handle the various effects of security challenges. The sufficient quantity and quality of available data is vital for the successful application of those IT solutions. The widely spreading sensor networks, social networks, and digital repositories allow us to access an extremely large amount of data through communication networks, and their volume is growing exponentially. Moreover, a significant part of this data is publicly available. On the other hand, there is a need for elastic, highly scalable e-infrastructures that can be built from mostly existing software and hardware components on demand to perform computational or data intensive analytics, simulations and predictive algorithms as fast as possible by the expert.

This paper presents some features and emerging trends of e-infrastructures and software tools based on the above requirements, which are capable of collecting and processing (open) data e.g. NoSQL database technologies, cloud computing, workflow-based and so-called orchestrator software tools.

Certain strongly related domestic, large-scale research and development projects are also discussed for illustration purposes as well as for reference, in which the Institute for Computer Science, Hungarian Academy of Sciences played a key role in the last decade. For example, the Agrodat.hu project is briefly described, aiming to process large quantity of monitored environmental parameters, images, non-structured documents (e.g. open social network data sources). The results of the Agrodat.hu projects are strongly interconnected with environmental and societal issues, or can easily be adopted.

Keywords: cloud computing, sensor data, unstructured data, IT platform, automatic deployment, scaling, workflow, Big Data, security, open source

Introduction: aspects of environmental and societal security involving electronic infrastructure

Environmental and societal security are two important areas in the generic topic of security issues that have a profound impact on citizens' daily lives and their standard of living. For example, if environmental security is reduced, regional or even global food shortages may occur. Reduced societal security often leads to the formation of parallel societies, which may cause problems in all aspects of the population's daily lives.

Numerous IT research and development achievements can provide efficient support and toolset for the security experts to prevent, detect, and handle the various effects of security challenges. The sufficient quantity and quality of available *data* is vital for the successful application of those IT solutions. The widely spreading sensor networks, social networks, and digital repositories allow us to access extremely large amount of data through communication networks, and their volume is growing exponentially, and a significant part of these data is public. On the other hand, there is need for flexible, highly scalable *e-infrastructures* that can be built from mostly existing software and hardware components on demand to allow experts to perform computational or data intensive analyses, simulations and predictive algorithms as quickly as possible.

In the following we will describe the characteristics and key trends of future-proof information technology infrastructures and software tools that satisfy the above requirements and are needed for the collection and processing of (public) data, and, by way of illustration and reference, a few closely related large-scale research and development projects of domestic relevance in which the Institute for Computer Science and Control of the Hungarian Academy of Sciences played a key role during a period spanning more than a decade to date.

Challenges: NoSQL-based storage of sensor data, unstructured data and other (public) data

More and more data is available due to digital services spreading like wildfire, and most of these data can be freely accessed through the Internet on servers run by various service providers. In addition to their volume, the diversity and heterogenous format of these data presents another challenge – it is worth using different methods for storing and processing text documents, data from social networks or values measured by sensors. The following introductory subsections will present four of the various approaches, with a special emphasis on their scalability and IT resource requirements in regard to cloud computing to be discussed later.

Text documents

Exceeding the limitations of the conventional relational databases *(SQL, Structured Query Language),* one of the most popular so-called NoSQL databases (Harrison, 2016) is the document-oriented MongoDB (MongoDB, 2015). Essentially it uses JSON *(JavaScript Object Notation)* and BSON *(Binary JSON)* formats that can be stored and processed more

efficient that the ubiquitous XML *(eXtensible Markup Language)*. It also supports setting up a distributed database on multiple servers if necessary, which provides good scalability thanks to the so-called replication and sharding techniques. Due to their architecture these systems may require significant data storage and network capacities in cloud computing.

Social networks

Graphs from mathematics are the best choice to describe connection network with edges, vertices, and their properties. Graphs can also be described using conventional relational databases, but querying such databases would be particularly resource intensive, and formulating the SQL requests themselves may prove to be cumbersome, because SQL lacks support for traversing graphs. In the past the so-called RDF *(Resource Description Framework)* and triplestores (Harrison, 2016) were used to describe connections. In the recent years more and more novel and popular solutions arose for the description of graphs, such as the open source Neo4j (Eifrem et al., 2015) – although it does not offer efficient support for splitting graph databases over several servers (see the above mentioned sharding technique), but it is still capable of storing and querying graphs with a billion vertices or more. Graph-based database solutions have significant resource requirements, mostly in terms of memory size, I/O (input/output) throughout and processing power in the cloud.

An agricultural precision project of MTA SZTAKI uses an HPE IDOL technology based solution to process newsfeeds and text documents received via social networking (see the details below).

Time series sensor data

Thanks to the fact that sensors consume less energy and grew smaller from day to day and to the ever increasing throughput of communication networks, sensor networks are gaining ground rapidly. Using a conventional row-oriented relational database (SQL) to store and process data captured from the sensors typically as a time series would provide a solution with very limited scalability due to the conventional structure in which the data is stored. A column-oriented approach allows for, among other things, reading sensor data in blocks, which is a must for efficient analytical and prediction algorithm execution. However, with a column-oriented storage solution data manipulation operations will take more time: usually this problem is tacked through the use of so-called delta storage devices or by batching write operations. For example, the column-oriented approach is used by the open source HBASE (Harrison, 2016) or the HPA Vertica (Agrawal, 2014) database too.

In its agricultural precision projects and Connected Car projects, MTA SZTAKI achieved considerable successes in research and development by using a cloud-based high availability scalable distributed Apache Cassandra (Hewitt, 2010) database (see the details below), targeting the processing of time series sensor data. Various implementations of column-oriented databases may require high data storage and networking capacities in the cloud.

Memory-based databases

The stringent requirements for data processing and evaluation usually prompt the designers of IT infrastructure to build a *real-time* system. One of the possible ways to achieve this goal is to move as many of the column-oriented NoSQL database tables as possible from disk storage subsystems with high response times to the operative server memory. Another factor that also supports this option is that column-oriented storage allows for extremely high data compression ratios, especially in the case of the above mentioned time series data (since it is enough to store the difference between the results of subsequent measurements instead of the values itself). Obviously, for memory-based databases, suitable mechanisms should be in place for regular synchronization of the operative memory and the mass storage subsystem in order to minimize the risk of data loss. There are two major systems that rely on memory-based databases and steal the limelight in their own respective areas: the SAP HANA platform (Harrison, 2016) and the open source Apache SPARK platform (Konwinski et al., 2015). Allocating the largest possible memory capacity to the so-called virtual machine running in the cloud is particularly important for memory-based databases – up to several hundreds of gigabytes of memory are available for such applications at public commercial cloud service providers.

MTA SZTAKI integrated the most recent release of Apache SPARK into its agricultural precision project (for the details, see below).

Solutions: properties, features, and current trends of cloud computing

Cloud computing was conceived with the basic idea of providing a cost-efficient, reliable, easy-to-access and scalable platforms that meet, among other things, the previously outlined various requirements even at recurring or occasional peak loads as necessary.

Characteristics of cloud computing

The definition of cloud computing was more or less finalized during the past decade; the most quoted definition (Sosinsky, 2011) came from the National Institute of Standards and Technology of the United States, and outlines the following key service criteria (see Figure 1).

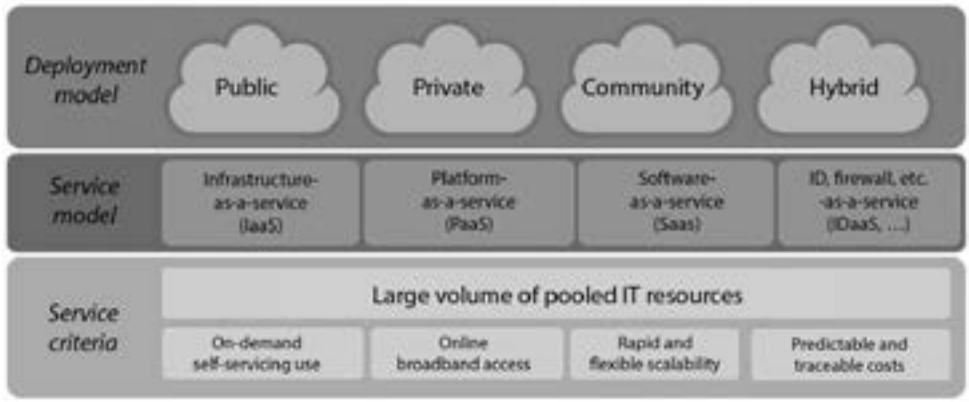

Figure 1

Deployment and service models and criteria for cloud computing

Source: Author's own contribution based on Sosinsky, 2011

- *On-demand self-servicing use:* The user of the cloud service can use either computing capacities (for a certain period of time) or storage space (e.g. within a certain quota). The process of requesting the service does not require intervention by the operating staff of the cloud service provider or organization: it is fully automated, and performed through a low-level programmable interface or a high-level (graphical or command line) user interface.
- *Online broadband access:* Cloud management services and computing, storage, network, and other capacities themselves can be used with the most popular standard access and high-level security protocols. This allows for accessing and exploiting cloud services remotely, using either a thin (mobile phone, tablet) or a thick (notebook, workstation) client.
- *Massive amounts of pooled information technology resources:* The information technology resources set up and operated by the cloud service provider company or organization usually service multiple users (clients) in a so-called multi-tenant model. In this model the available massive amounts of physical and virtual resources are dynamically reserved, allocated to the given user according to the received and automatically processed requests, and then released. Physical resources providing a given capacity requested by the user or, in certain cases, implementing a high-level service are reserved in large-scale clouds without regard to their location, if there is no constraint on geographic position. Due to legal and security considerations, however, users can specify the high-level location of the physical resources, for example, the region, the country or the data centre hosting them. A few examples of pooled physical resources: storage space (disk subsystem), processing capacity (multi-core CPU and GPGPU, which means servers based on general purpose graphics processing units), operative memory, high-speed network.
- *Rapid and flexible scalability:* While operating a given service or running applications, computing, storage, and other capacities can be flexibly allocated and released, even

automatically so that the service may be scaled or the running application may have access to more resources before the given resource limits are reached, in order to meet the ever changing needs.

- *Predictable and traceable costs:* In a multi-tenant system the cloud can automatically manage the pooled resources to optimize their utilization and make it as cost efficient as possible. In order to achieve this goal, quantifiable metrics characterizing the use of services by the individual users must be monitored and logged, including, for example, the size of storage space, processing (CPU) time, and network data traffic. The use of resources can be monitored and tracked (booked) by the cloud service provider in regard to the users.
- Service models

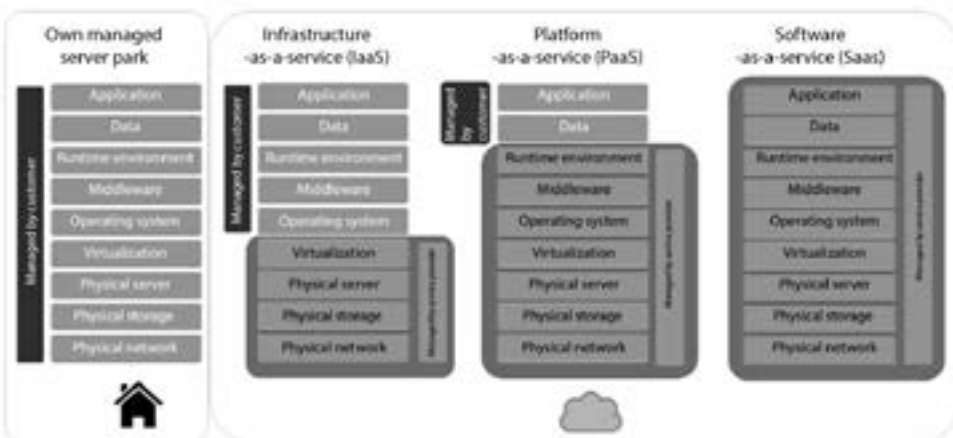

Figure 2

Comparison of service models

Source: Author's own contribution, based on Farkas et al., 2013

Compared to proprietary corporate information systems, a given cloud service may belong to three distinct categories according to the *service model* it implements (see middle row in Figure 1 and Figure 2). It is easy to see that proceeding from left to right the service provider manages more and more layers in all services models, and can hide in this way low level details of these layers from the cloud users:

- infrastructure-as-a-service (Iaas);
- platform-as-a-service (PaaS);
- software-as-a-service (SaaS).

ID (identification), firewall and VPN (virtual private network), etc. as security service are becoming more and more popular: considering the underlying concept, they can be interpreted in the platform-as-a-service or software-as-a-service model.

Deployment models

Possibilities of the various *deployment models* (upper row in Figure 1) (MTA Cloud):

- *Private cloud:* The most important feature of a private cloud is that it is operated for the exclusive use of a single company or organization, and typically only their registered and authorized employees have access to it. The emphasis is on "exclusive use" as the private cloud may be owned and maintained by the company or organization itself or even a third party, and the information technology resources may be located on the premises of any of these parties.

Note: Conventional data centres can be transformed into private clouds, but such a move requires, among other things, virtualization and automated resource management based on a suitable methodology, and furthermore a portal that supports self-servicing. It is also necessary to measure the use of resources. The in-house cloud of MTA SZTAKI is a private cloud that has been operating since 2012.

- *Public cloud:* In stark contrast to a private cloud, a public cloud can be defined as an information technology environment established for serving public business or private users. The owner and operator can be a company, an institute of higher education, or a government agency, and the physical IT resources may be hosted on its premises. An example of such a cloud is Amazon AWS and Microsoft Azure, both regarded as a pioneering solution in their own areas (Sosinsky, 2011).
- *Community clouds:* When examined from *the users' point of view*, community clouds appear as an interesting transition between private clouds and public clouds. As an example, we can point to the MTA Cloud established in 2016, which involves a federated cloud deployed and operated by two organizations of the Hungrian Academy of Sciences, Wigner Data Center and MTA SZTAKI, at both sites, serving the research community of the academy (see Figure 3), including research projects pursued by other organizational units of the operators.

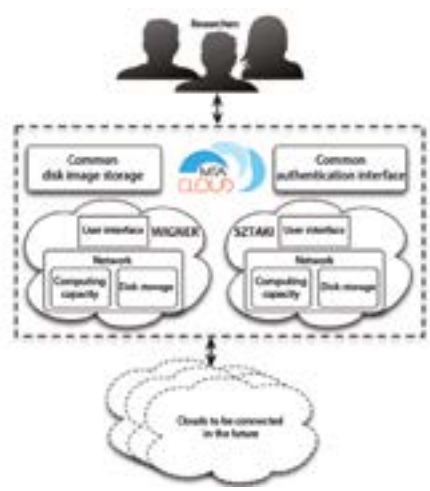

Figure 3

An example of a community cloud service

Source: Based on MTA Cloud, revised version

- *Hybrid cloud:* Considering its *physical IT resources*, a hybrid cloud is a result of a special combined use of a private and a public cloud. While both clouds (public and private) retain their independence, but through the use of network connections and suitable protocols, they can transfer (escalate) data and/or application between them as necessary. The most frequent applications of hybrid clouds are summarized as follows:
  - Requesting additional capacity from the public cloud: In such cases, even though the resources of the private cloud could grant the incoming requests to a limited extent, but if the load approaches or even exceeds the given upper limit, then new resources are allocated in the public cloud. Then, after connecting the new capacities to the resources in the private cloud, the public cloud will start processing requests constituting the extra load.
  - Segmentation of data: The currently applicable privacy policies and other security rules or regulations may restrict the scope of data that can be stored in the public cloud, including on servers located in other countries. In such cases, it is unavoidable that the data (and the services processing them) should be segmented as stipulated by the legislative environment: data regarded as sensitive will be stored and processed in the private cloud while data not subject to any restrictions will be processed in the public cloud.

Open source cloud platforms

Two open source cloud computing platforms and their various distributions have become the most popular: OpenNebula (Moreno-Vozmediano et al., 2012) and OpenStack (Radez, 2015). OpenNebula emerged from a European initiative, and due to its architecture, which establishes direct synchronous secure (SSH) connection between the nodes of the infrastructure, it is mostly recommended for building small and medium-size infrastructure-as-a-service (IaaS) clouds. Thanks to its low resource requirements and ease of deployment, it is very popular in the educational and academic sphere. In addition to the essential IaaS functions such as the management of users and virtual machines, it also supports the so-called *cloud bursting* feature (e.g. by using the hybrid cloud model approach) through the EC2 *(Elastic Compute Cloud)* (Sosinsky, 2011) interface that became a *de facto* standard for accessing other cloud resources if additional capacities are needed.

OpenStack started as a joint initiative of the National Aeronautics and Space Administration (NASA) and the company Rackspace, and by now it has become the most significant open source IaaS platform. On the one hand, it has an extremely large user and developer community, and a number of global enterprises chose to support this solution and have been using it for their own goals. On the other hand, its architecture, which is modular and uses asynchronous connections between nodes, allows for building even large-scale IaaS infrastructures. Also, it supports PaaS and so-called *bare metal* (e.g. GPGPU subsystem) provisioning. One of the leading OpenStack distributors is the company Mirantis and HP Enterprise with its Helion (HPE Helion Documentation) brand.

Commercial cloud service providers

The most recognized and the largest service providers include Amazon, Microsoft and Google (Sosinsky, 2011), but there are several service providers which are specialized or have a larger market share outside Europe that are worth mentioning here, such as Rackspace or Salesforce.com. By launching the AWS cloud services, which was driven by the effort to utilize unused information technology resource in the international enterprise's server centres, Amazon pioneered and shaped the market in the IaaS segment. Microsoft became a key player in the PaaS segment as its development systems can directly leverage platform services provided by Microsoft Azure (Farkas et al., 2013), which makes it easy for the developed software solution to benefit from its advantages. For many reasons Google is seen as the dominant market player in the SaaS segment: its mail (e.g. Gmail) and file sharing (Google Drive) services are also based on cloud computing.

NoSQL in the cloud

Cloud computing offers various methodologies and options for large-scale NoSQL-based data storage solutions. A frequently encountered basic architecture involves the use of some sort of NoSQL database in a distributed configuration with several virtual machines. Most of the PaaS service providers support such implementations, and thus, for example, the above mentioned Cassandra and MongoDB database are also available as part of Amazon's public AWS cloud services. In the case of distributed databases a special emphasis must be placed on fine tuning the consistency level: practical advice is available in that regard from several sources (Harrison, 2016; MongoDB, 2015; Hewitt, 2010).

Deploying a Hadoop service in the cloud

In connection with NoSQL based analytical tools there is a profound need for supporting MapReduce (Harrison, 2016) applications, the most popular implementation of which is the open source Hadoop. Now almost all cloud platforms and service providers offer support for such processing tasks: for example, Microsoft Azure uses the HDInsight brand name, while Amazon AWS and OpenStack call it Elastic MapReduce, and Google introduced the name DataProc. In OpenNebula the application is available on the marketplace in the form of Hadoop virtual machines.

There are already cloud service provider independent solutions that can use any private clouds: now the WS-PGRADE/gUSE (Kacsuk, 2014) system and Occopus (Kecskeméti et al., 2014) together can start Hadoop clusters in the clound and integrate them into complex workflows. For further details, see below and refer to the website of Occopus (Occopus project).

Designing complex scalable IT platforms for applications of public data for security purposes

Requirements

The following is a non-exhaustive description of a few of the key characteristics of a system that is potentially suitable for storing and processing public data in an advanced, cost-efficient way for security applications:

- Essentially, due to the sensitivity of the derived results, it is a *private or community cloud*, which is also suitable for hybrid operation as necessary.
- Support for not just MapReduce but also for *workflow based* solutions for data processing to promote simple use in order to reduce the workload of analysts and security experts.
- Wherever possible, *open source based* software elements are used due to economy (reduction of investment costs) and transparency related considerations.
- *Automated so-called "orchestration"* and *management tools* to optimize human resource costs of IT operations.

In the following, we will present four solutions related to the above four requirements, with a few actual implementation details for some of them.

Design of data, installation and workflow – orchestration

There are several techniques and software tools to describe data, installation and workflow processes in the scientific domain, such as Kepler (Ludäscher et al., 2006), and in the area of business applications, such as WS-BPEL (Vasiliev, 2007).

In both cases these descriptions allows for the use of acyclic directed graphs in order to create a graphical definition of the schedule of various interrelated activities (algorithms or even cloud-service deployment), the dependencies between them, followed by the execution of the produced complex processes (analyses, simulations, service provisioning) on the designated information technology platform, using a suitable data and workflow manager or infrastructure orchestrator. Such systems offer richer options to describe complex scalable solutions, including *parameter sweep,* and many of them support the design and coordinated use of various cloud-based computing platforms, such as the combination of the gUSE/WS-PGRADE (Kacsuk, 2014) workflow system developed by MTA SZTAKI since 2003 and the most recently developed Occopus (Kecskeméti et al., 2014) infrastructure orchestrator. Of tools designed for similar uses OneFlow within OpenNebula and TOSCA (Wettinger et al., 2013), which is favoured by the academic community can be mentioned as typical examples.

It is important to note that expectations regarding workflow-based description were already expressed at higher professional policy-making levels as mentioned in a report released by the European Union in 2010 (High Level Expert Group on Scientific Data, 2010). Working with scientific data, the expert group outlined in its report a framework for a collaborative data infrastructure that could be used by the EU to utilize the ever increasing

amounts of data produced. It is easy to see that the workflow appears explicitly at two levels in the framework: on the one hand, in the layer of support services through the generation of workflows, and on the other hand, in the combined shared data services through the execution of those workflows.

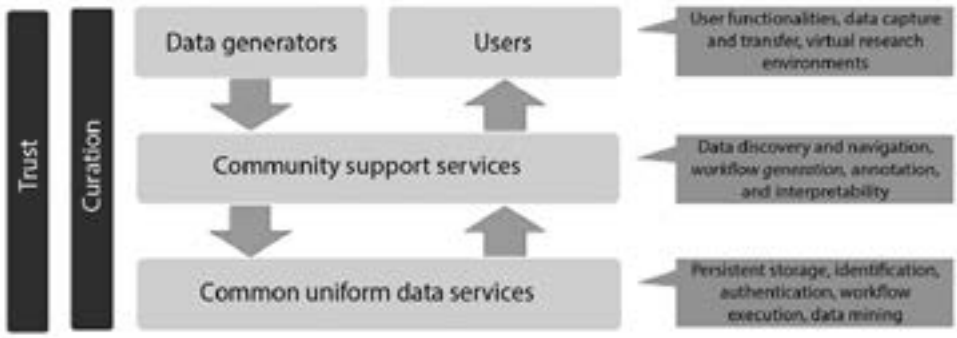

Figure 4

Role of workflow in collaborative data infrastructure framework

Source: Based on High Level Expert Group on Scientific Data (2010), revised version

Workflow description using gUSE/WS-PGRADE in the agINFRA project

gUSE *(grid and cloud User Support Environment)* is an open source workflow based so-called *science gateway* framework (Kacsuk, 2012) developed by the Laboratory of Parallel and Distributed Systems, Institute for Computer Science and Control, Hungarian Academy of Sciences. Its web interface, WS-PGRADE (see first row of Figure 5), which is available to registered users, offers user-friendly access to the development of distributed workflow based applications, and allows for executing them in clouds, in distributed computing infrastructures (e.g. grid) or in computer clusters. The gateway provides an interface between the expert and the distributed computing infrastructure that the expert intends to use. It is achieved through a combination of the benefits of the underlying technologies, including both front-end and middle-tier services, which are combined to build a gateway.

It is important to emphasize that the system is not dedicated to a specific area of application, which means that many other areas can leverage its benefits, which can be customized. Application developers have access to advanced workflow support services (abstract and actual workflows, templates, applications, and projects), which enables them to develop new workflow-based applications that can be uploaded to the gUSE application repository where common users can easily access and run them using the workflow-interpreter and the other middle-tier services (see the middle row of Figure 5). In this way, experts can focus on the meaningful part of their work instead of being bogged down by having to familiarize themselves with the specifics of various cloud-based infrastructures or maybe even creating them; the task management and data management tier support several popular distributed

information technology platforms and data storage technologies (see bottom row of Figure 5) through the DCI Bridge and Data Avenue components (Kacsuk, 2014).

The key objective of implementing the framework system was to make computing and storage IT resources easier to use for the end users in this way, regardless of whether they are in an OpenStack or OpenNebula based cloud or any other popular distributed computing infrastructures.

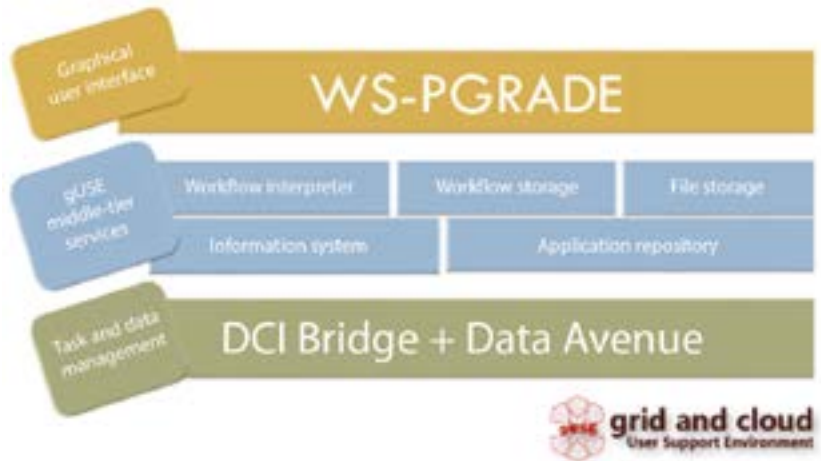

Figure 5

High level architecture of gUSE/WS-PGRADE

Source: Based on project EU FP7 SCI-BUS, revised version

The gUSE/WS-PGRADE system was added to several international projects of the European Union in more than thirty major fields of applications and communities (Kacsuk, 2014). The most important gUSE/WS-PGRADE solution-based project was SCI-BUS coordinated by MTA SZTAKI under the EU FP7 programme where the CloudBroker platform developed and adapted to gUSE became the first out of more than five hundred evaluated innovative solutions that were delivered as part of the 7th Framework Programme for Reseach and Technology Development, the Competitiveness and Innovation Framework Programme (CIP), and the Horizon 2020 programme of the European Union (De Prato et al., 2015).

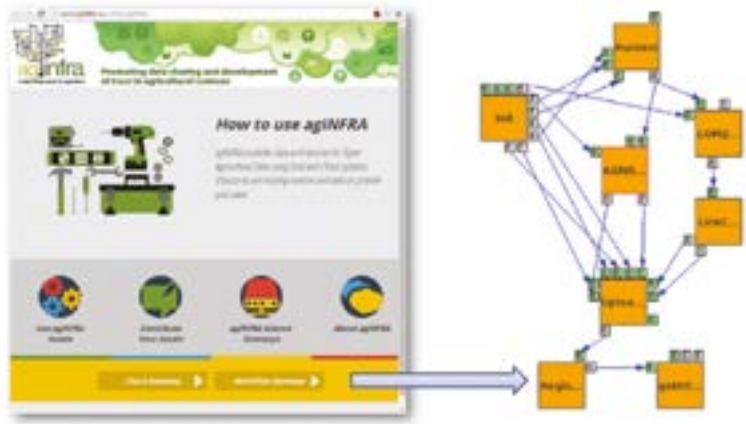

Figure 6
agINFRA science gateway and demo application: aggregation workflow

Source: The author's own contribution

Another key application under the EU FP7 programme was the agINFRA project where a specialized science gateway, a special portal (see left side of Figure 6) by customizing the gUSE/WS-PGRADE solution, and an *aggregation* workflow (see right side of Figure 6). The implemented solution (Balaskó et al., 2014), collects, among other things, documents and metadata to be indexed in the graph by using the CIARD RING metadata store and a reference collection pointing to international data warehouses, libraries, and repositories that focuses on documents of agricultural relevance (see the orange-coloured *Harvest* node), and performs data conversion and (e.g. *LOM2* and *AGRIS* nodes) checks as necessary (*Linkc* node). Dependencies and required data transfer operations between activities – i.e. nodes – in the workflow are defined by blue directed edges as shown in the figure, and with small numbered components.

Occopus infrastructure orchestrator

First of all, it is important to note that orchestration is more than simple automation because it integrates, for example, individual automated tasks into workflows. For example, a computer network orchestrator program typically checks the virtual local area networks (VLAN), selects the most capable one, then analyzes the network *switches* to identify the ones where changes are needed. Using the collected information, the program generates the required configuration settings and transmits them to the targeted active network devices. Then the program will continue to monitor them so that any issues that might arise may be corrected. This means that the orchestrator device combines and, depending on the circumstances (or changes thereof), coordinates the automated actions: in our example, this includes all phases from mapping the network, through configuring devices and applications, to managing the network. In short, it also controls the complex process of configuration management.

These days most of the large enterprises and organizations use server virtualization and cloud computing, which promotes the adoption of the orchestration methodology. Many developers have turned their attention towards automation and orchestration recently as organizations are forced to invest a lot of time and human resources into rolling out and commissioning their information infrastructures (including automating the installation of operating systems and applications, performing storage and network configuration, fine tuning firewalls, etc.) and into the maintenance of the deployed systems. As a result, orchestrator developers aim to create software that helps or perform on its own these often repetitive tasks that are so time consuming for humans.

When commissioning servers in an electronic infrastructure, coordination of the services and the configuration work required to accomplish it may consist of hundreds of complex steps that must be completed before a single server is put into operation. Installation and coordination of applications require complicated design and project management efforts for each component (server, storage space, virtualization, network, and security). However, regardless of the project at hand, commissioning involves progressing through a similar series of steps, even in the case of relatively simple solutions. When working with IaaS type clouds, the number of virtual servers and other singular resources may amount to thousands. Setting up, configuring, and modifying them would take a long time, but there are already orchestrator tools available to manage these tasks, allowing for the turn-key deployment of complex/complete infrastructures with minimum human involvement.

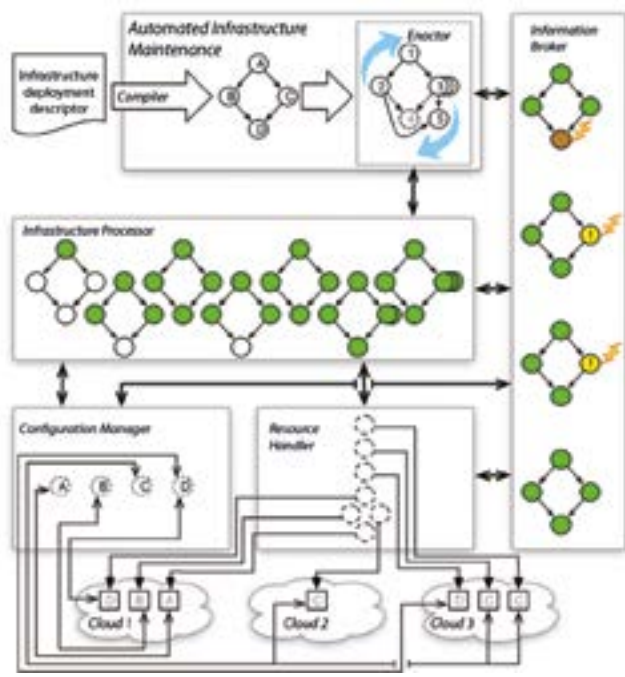

Figure 7

Internal structure of the Occopus hybrid cloud orchestrator

Source: Based on Kecskeméti et al., 2014 (revised version)

Figure 7 illustrates the main components, connections, and operations of the Occopus hybrid cloud orchestrator developed by MTA SZTAKI (Kecskeméti et al., 2014) from the infrastructure operator's point of view. The Occopus solution consists of five key components:

1. Automated Infrastructure Maintenance;
2. Infrastructure Processor;
3. Resource Handler;
4. Configuration Manager;
5. Information Broker.

Automated Infrastructure Maintenance

The automated infrastructure maintenance component is the only one that has a full view spanning all the parts of the operated infrastructure. It is also connected to the external world, and essentially performs two main tasks: on the one hand, it orders the execution of virtual infrastructure requests, and on the other hand it destroys existing but no longer needed virtual infrastructures in the cloud.

Its input is an *Infrastructure Deployment Descriptor,* which includes information necessary for building the virtual infrastructure: the types of infrastructure nodes, their dependencies, rules for error handling, such as handling management errors or over- or underestimated allocations. When the interface receiving the request get the descriptor, it is converted into an internal representation, see *Compiler* in Figure 7. If there is a compilation error, then the component will provide feedback, otherwise the process will continue.

The *Enactor* subcomponent is a basic part of the system: while building a virtual infrastructure, it sends the infrastructure processor requests related to nodes according to the order derived from the dependencies. Once the order is sent and the requested virtual infrastructure is completed, it will monitor the state of the infrastructure all the time in order to detect errors and it will resolve those according to the rules included in the infrastructure deployment descriptor.

These rules determine the activities that need to be completed, such as node redeployment, re-estimation of dependencies, if one of the nodes of a given type becomes inaccessible. For example, if the D-type (no. 4) node in the box of the automated infrastructure maintenance component shown in Figure 7 fails, then it will be substituted by node no. 5. Unfortunately simple rules cause so-called oscillation, and for this reason the enactor uses complex rules to eliminate the issue.

The enactor maintains the virtual infrastructure completely on its own, except if changes are needed in the infrastructure deployment descriptor. In this case, as the first step, the infrastructure maintenance component updates the descriptor, and then the Automated Infrastructure Maintenance component compiles the new internal representation of it, and finally the enactor switches to a transitional state. In this state, the enactor evaluates the differences between the old and the new internal representation and if it finds only new error handling rules, then the enactor will adjust them to the infrastructure; for example, if a new scaling rule calls for fewer instances under the same load, then the extra instances will be released through the infrastructure processor, and the enactor will revert to normal operation. If the evaluation finds new node types and dependencies, then the currently

operated virtual infrastructure will be reorganized according to the new infrastructure deployment descriptor.

Finally, when the virtual infrastructure is to be destroyed, the enactor will send destruct requests for all previously created nodes to the infrastructure processor. The order of destruction is reversed compared to that of creating the nodes, and in this way the dependencies of all node types are gradually dissolved.

Infrastructure processor

With the infrastructure processor, Occopus implements a new level of abstraction over the virtual infrastructure. As we saw in the previous part, the infrastructure processor receives note construction or destruction requests from the enactor. When the first node construction request is received for a virtual infrastructure, this component will create a so-called administrative group for the future virtual infrastructure. Nodes can share information among each other through this administrative group: for example, new nodes will get their dynamic properties (e.g. IP addresses) from the existing nodes.

Node construction requests are processed as follows: first, the infrastructure processor confirms that the configuration manager knows what type of node is to be constructed. As soon as the node type is known, the infrastructure processor virtual machine sends the request context to the resource handler component. Into the contextualized information, the infrastructure processor inserts a reference to the previously created administrative group and the expected node type of the future virtual machines.

In Figure 7, in the box representing the infrastructure processor, the process of construction can be seen from left to right, starting with the initial step and ending with the final state. The solid green circle indicates the already processed steps of infrastructure deployment. On the other hand, node destruction requests are forwarded by the infrastructure processor directly to the resource handler. When the final node is destroyed in the virtual infrastructure, then the infrastructure processor will also destroy the associated administrative group too.

Resource handler

The core function of the resource handler is to create a new general abstraction tier over the IaaS functions of the clouds, allowing for the creation, monitoring, and destruction of virtual machines. With the function comes a *plugin based* architecture, which can be implemented on most of the IaaS interface. At this time, Occopus supports OCCI, EC2 (Amazon), Nova (OpenStack), CloudBroker and Docker interfaces (Occopus project). These plugins are expected to carry out incoming simultaneous requests as soon as possible.

In order to improve the performance and flexibility of the deployed virtual infrastructure, the resource handler also provides a virtual machine scheduler, which works even across several clouds. The scheduler function of the virtual machine allows either the infrastructure operator or even the user initiating the virtual infrastructure to define the cloud selection criteria.

In our example, the order of the incoming virtual machine requests is shown by dotted circles in the resource handler box of Figure 7: the first one is at the bottom, the last one is at the top, while parallel requests are shown side by side. Requests to assign a certain virtual machine to a given cloud ("Cloud to VM") are marked by arrows, with little squares representing the actual virtual machine in the cloud. Every virtual machine shows the contextualized node types in grey letters (from A to D).

Configuration Manager

The configuration manager component manages the installed software and its configuration at node level, and a lot of widely used tools are available to that end. Component interfaces are provided, for example, for Chef, Cloudify, Docker, Puppet, and SaltStack. These applications use custom node type definitions (e.g. in Chef, they are called "recipes", in Puppet "manifests"). With already defined types of nodes, the configuration manager allows for the reuse of those definitions, even from external sources, and thus the conventional node type definitions will only serve as references to the custom definitions. Also, in the infrastructure deployment descriptor new node types can be defined in the extended node type definition. In this way, the definitions allow the configuration manager to select a suitable underlying node management tool: for example, if a "recipe" is used to describe the type of node, then Chef will be used.

Again, in our example shown in Figure 7, node type descriptions are represented by dotted circles within the configuration management component. Arrows between virtual machines and type descriptions indicate the connections between the virtual machines and the configuration management component as it receives and applies node type descriptions. These operations ensure the correct configuration of the software components needed for the virtual machines so they may play their role within the virtual infrastructure operated by Occopus.

Information Broker

The Information Broker component is designed to ensure that Occopus can make well-founded decisions in regard to the state of the requested virtual infrastructure (e.g. it can reduce redundancy, etc.). This is ensured by the information broker with two functions: request transformation and information aggregation. The first activity involves the information broker transforming sometime abstract or conceptual requests into actual information units that can be accessed by the various Occopus components and underlying clouds: for example, the request concerning the load of the D type node shown in the figure can be transformed into the CPU utilization of a virtual machine in Cloud 1 or Cloud 3. The second operation, information aggregation, takes place when the information broker receives requests that only contain composite information. In such cases it forwards the requests to each relevant Occopus component, and, if necessary, to the virtual infrastructure too. After receiving the responses from the components, the information broker calculates the aggregate value of the responses and returns it as a response to the original request.

In the example shown in Figure 7, the following is shown from top to bottom in the information broker box:

- State 1: At regular intervals the enactor sends a request to the information broker to ascertain the availability of nodes, and the information broker forwards this request to the nodes. The component does not receive an answer from the D type node, and as a result it declares the latter unavailable (the green circle representing the D type node is crossed out in red), which renders the virtual infrastructure unusable. For this reason, the enactor sends a new D type node construction request to the infrastructure processor.
- State 2 and 3: The C type node is under an ever increasing load. If the load is excessive (the green circle representing the C type node will turn yellow and an exclamation mark will indicate excessive load) and the given virtual machine can no longer handle the expected load, then the enactor will increase the number of C type nodes, thereby reducing the load on the individual C type nodes.
- State 4: All nodes types are working properly.

Reference: Agrodat.hu project

In the following we will describe a Hungarian reference platform that offers a solution for sensor and image data processing and thematic processing of social media and open document repositories, mostly based on open source cloud computing and big data. It can be adapted extremely easily not only to environmental but other security related applications as it was developed according to, among other things, the above defined requirements.

Figure 8

High level overview of the Agrodat.hu project

(1) data sources: sensor networks, international data warehouses (top) and (2) research platform: analytical and decision support software elements, big data server park (bottom)

Source: The author's own contribution

As part of the Agrodat.hu project, MTA SZTAKI designed and implemented a regionally unique and exceptionally large scale research infrastructure aimed at promoting precision agriculture. Precision agriculture is meant to reduce risks arising from decisions made during agricultural production and from changes in the external environment, allowing for cutting back on the use of fertilizers and plant protection products, and achieving sustainable growth of the average yield. In this way, significant improvement can be realized in the area of environmental security.

By adapting the big data and cloud technology based analytic, forecasting and decision support framework to agricultural applications, the newly developed platform enabled the creation of a constantly extended knowledge base with an ever wider scope for the agricultural segment. The research group enabled the system to use a NoSQL (Cassandra) based solution (Hewitt, 2010) to reliably and efficiently collect structured time series, image and other data from a huge number of sensor columns installed in the field by partner companies, and to efficiently exploit information contained in those data with the assistance of agricultural experts, using Apache Spark (including, among other things, Hadoop) (Konwinski et al., 2015). Also, as a result of the agINFRA cooperation between the institute and the Food and Agricultural Organization of the United Nations under the FP7 programme (Balaskó et al., 2014), and with the involvement of additional industrial partners such as HP Enterprise and its strategic partner, non-structured data of international open data warehouses and community sites of agricultural relevance can be intelligently searched and integrated into the knowledge centre.

Collection of sensor data using a scalable NoSQL approach

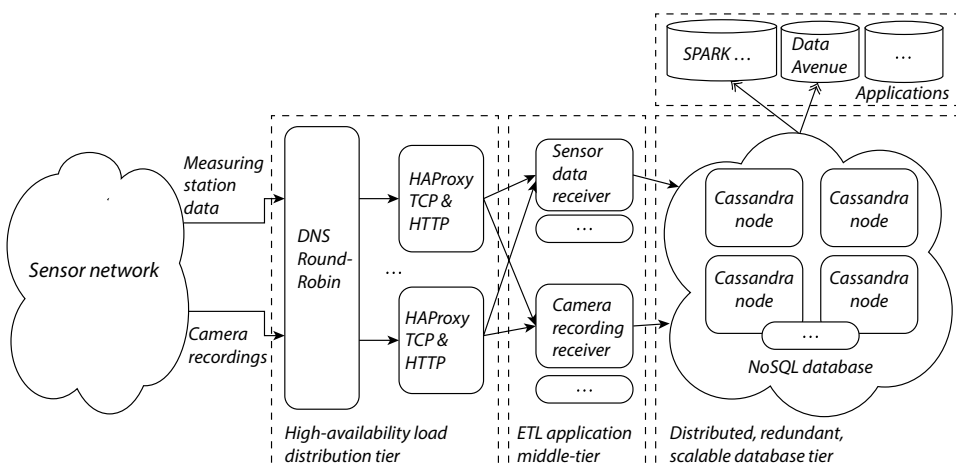

Figure 9

Agrodat.hu data collection platform with additional NoSQL data storage solution and other applications

Source: The author's own contribution

The first stage of receiving data from sensor networks is a high availability load distribution tier where a *round robin* algorithm is used to direct sensor data towards high availability *proxies* via HTTP/TCP protocol based communication. After this stage, the next tier contains redundant, sensor data dependent data pre-processors – so-called data receivers – in the ETL *(Extract, Transform, Load)* middle tier, which forward data for storage to the next tier, the distributed NoSQL database (Harrison, 2016). In the database tier, an Apache Cassandra based (Hewitt, 2010) NoSQL cluster was set up, which, on the one hand, is capable of storing huge volumes of input data quickly (basically, it handles database write operations), and on the other hand, it can efficiently perform the mostly read type database operations of the top-tier analytical (Apache SPARK [Konwinski et al., 2015]) and data transferring subsystems (Data Avenue, [Hajnal et al., 2015]), thanks to the support of consistency level tuning in Cassandra. It is important to note that as a result of this project, the data collection and storage infrastructure shown in Figure 9 can be deployed automatically and in an orchestrated manner through the use of the above discussed Occopus (Kecskeméti et al., 2014) solution in, for example, an OpenStack based cloud (Radez, 2015).

Intelligens search using IDOL

As part of the Agrodat.hu project, a service was developed to aggregate and organize non-structured data from thousands of various digital archives and other sources dedicated agricultural topics, including scientific papers, Facebook entries, educational materials, and other reports, and to make them searchable in an intelligent way, using open source as well as closed source software. The core of the system is a version of the HP Enterprise IDOL (HP Enterprise) search engine *(enterprise search engine)* licenced for 5 million documents. The data collection method (ETL), the user interface *(frontend),* and the agricultural dictionary that serves as an ontology are based on open source solutions and academic development efforts, thanks to the cooperation between ICSC and the Food and Agricultural Organization of the United Nations in the past.

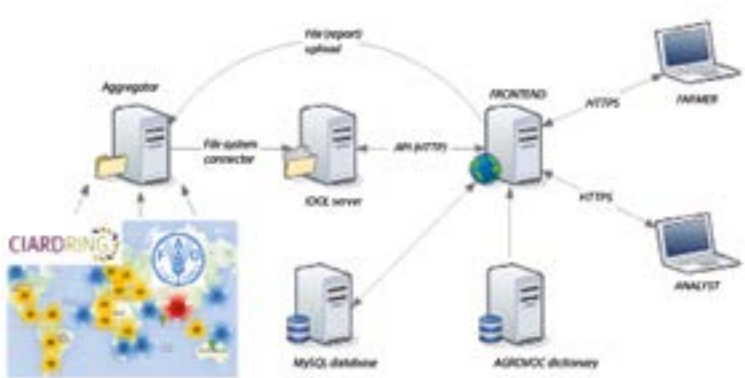

Figure 10

Intelligent search feature using HPE IDOL server and Agrovoc dictionary

Source: The author's own contribution

The solution developed as part of the project (see Figure 10) allows farmers and analysts working in the agricultural sector to explore relevant information. Access to information is provided by a *frontend* application that can be accessed with a web browser. The key function of the frontend is to expose the intelligent document search feature to the users, with additional administrative functions as necessary. Information needed by the frontend is delivered from two sources: one of them is a relational MySQL database that store conventional structured data needed for operation, and we also have access to a HPE IDOL server which stores and indexes unstructured natural language text documents. Indexes of the IDOL server are built from documents of various formats that are located at pre-selected locations in the file system. This means that the file system is one of the interfaces to this solution. Moving files from repositories in CIARD RING to the workspace is carried out by an *aggregator* software, which is based on the aggregator that was developed as the already described gUSE/WS-PGRADE workflow developed for agINFRA.

At the bottom of Figure 10 is another element: *Agrovoc* (Balaskó et al., 2014) is a dictionary that contains agricultural terms and phrases in 22 languages, with more than 30,000 important phrases of the agricultural jargon, and is capable of interpreting various relations between the expressions, such as synonyms and root terms (i.e. its data structure corresponds to a graph). The dictionary is available in several formats, of which we prefer the RDF format as we have access to open source solutions to read and query files created in this format. The system uses *Agrovoc* to offer, among other things, reasonable and accurate recommendations to users as they compose their search expressions, even in a different language. The system is also capable of processing data from social media sites: listing of relevant hits and similar functions work in the same way as in the case of data obtained from CIARD RING (Pesce et al., 2011) using the aggregator.

Figure 11 shows the search interface, which uses *Agrovoc* to offer accurate search term recommendations in multiple languages, for example, when entering the keyword *maize* or *kukorica* (see the drop-down list boxes at the bottom on the left and in the middle at the top of Figure 11). The search for *maize hybrid* returns the list of hits that is show in the middle of the figure, together with the list of the recommended related subjects on the right side. In response to a search for the word *disaster*, the system will list hits from agriculture related Facebook community newsfeeds (see upper right corner of Figure 11). For every hit list, we will get a relevance indicator estimated by IDOL, and it is also possible to refine searches using various logical expressions, even by defining the maximum distance between the occurrences of the given keywords.

Figure 11

The intelligent search engine interface of Agrodat.hu with illustrations

Source: The author's own contribution

The cloud-based Big Data infrastructure of Agrodat.hu

Commissioning of servers and information technology equipment needed for the operation of Agrodat.hu research infrastructure in the live environment was completed in stages using Hadoop, GPGPU and blade servers, and a particularly fast data storage subsystem (3PAR), which were integrated on the above mentioned OpenStack open source cloud platform, using Infiniband and a 40Gbps network.

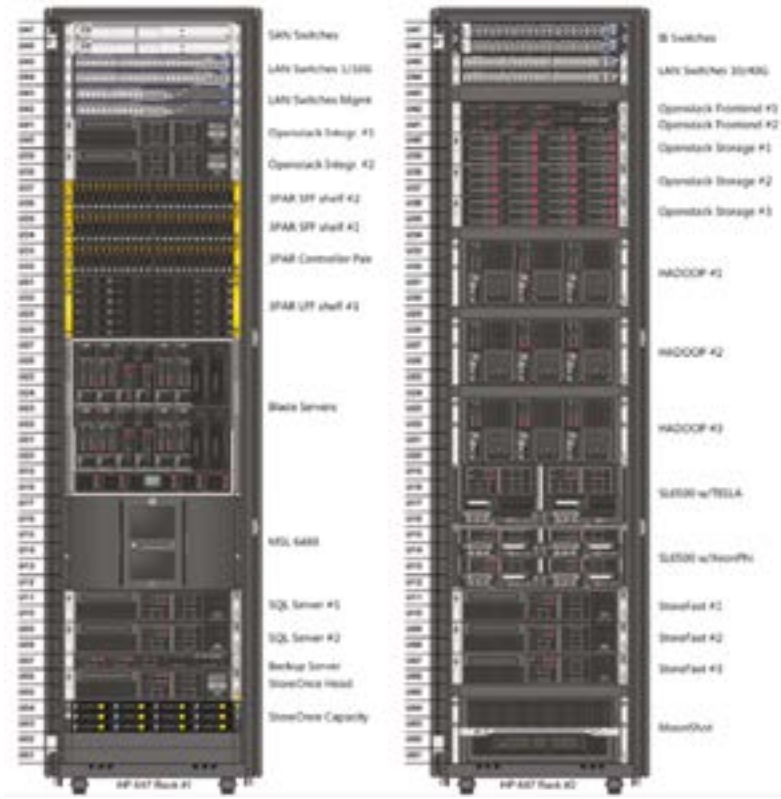

Figure 12

Research infrastructure – hardware configuration of the big data server centre of Agrodat.hu in two rack cabinets (network components, storage units, blade servers, database servers, tape units, cloud components, Hadoop [big data] and GPGPU servers, etc.)

Source: The author's own contribution

Directions of further development: the cuvée strategy

The cloud based big data platforms are gaining ground rapidly in the various application segments where the amount of data to be processed is undergoing an explosive growth. In this area, solutions can be divided into two major groups: *open source* and closed source *(proprietary)* platforms. Open source allows for, among other things, reducing investment costs and vendor dependency. With proprietary systems, the same could be achieved by using public cloud based on-demand services, however, it is often not feasible for big data systems due to, for example, the sensitivity of data or the exorbitant costs of data transfer. It should also be mentioned that if open source solutions are used, the availability of acceptable support and maintenance services, and the selection and integration of compatible software products may cause problems.

Among the potential new users of cloud-based and big data systems in the so-called *long tail* stage there are small and medium enterprises and start-ups, which are innovative but not so well-capitalized, and there are the small and medium-sized research laboratories and organizational units of the state owned institutions (universities, academies, public administration, home defence, etc.), which often prioritize open source, but fail to use the proper methods to build their IT infrastructures. As a result, the competitiveness of their R&D and other core activities is considerably reduced. In order for these segments to leverage the benefits of the cloud and big data – with the lowest possible cost of entry and vendor dependency – an approach that combines the best of both worlds should be developed and deployed as a platform: essentially, this is more or less the gist of the *cuvée* principle.

In connection with research efforts related to such a new approach, a successful *pilot* operation was launched as part of the Agrodata.hu project at the time of deploying the big data based and agriculture oriented (precision farming) knowledge centre and research infrastructure. Medium-term plans call for using the new approach, in addition to data obtained from the agriculture and UAVs, to meet the requirements of other fields, based on the accumulated experience. The targeted audience includes Industry 4.0 players, e-healthcare, automotive industry, telecommunication, and financial service providers as well as fields closely related to security (e.g. chemistry, meteorology, environmental protection and national defence), which, as a result, will be provided with increasingly accessible and efficient cloud-based support, specifically by performing simulations, real-time analyses and other mathematical modelling activities on a Big Data platform.

Achieving this goal is furthered by a stable technological foundation and knowledge base thanks to the fact that, due to its above mentioned reference projects, MTA SZTAKI became one of the dominant players of the region in the field of migrating workflow based large scale and complex research applications to distributed computing (EU FP7 SCI-BUS project), cloud-based (EU FP7 CloudSME project) and self-organizing, volunteer platforms (EU FP7 IDGF-SP project). Meanwhile significant steps were taken in the institute in the research of one of the cornerstones of the new research approach: the interchangeability, suitability for system integration and interoperability of e-infrastructure components (EU FP7 SHIWA project). The targeted research project builds on the above results and the development efforts of the recently launched *One Click Cloud Orchestrator* (OCCO) (Kecskeméti et al., 2014) and *Data Avenue* (Hajnal et al., 2015) projects. As such, the platform to be built will be able to work with several large component suppliers, and even automate and orchestrate the automated instantiation and/or accessing of their proprietary big data components (Harrison, 2016) (NoSQL, stream based processing, Hadoop, machine learning, etc.) as well as their integration, scaling, and operation with open source software solutions.

As a key benefit, the new *OCCO* generation, the *Occopus* (Occopus project) will, in the long run, enable orchestrated deployment and operation of a big data research platform in either a private, a public or a hybrid cloud that is scalable depending on the load, taking into account the user's budget and actual requirements (e.g. sensitivity of data). All this is to be achieved, with as little human intervention as possible, by combining the benefits of open source and proprietary solutions and ensuring the lowest possible level of vendor locking.

References

Agrawal, Rishabh (2014): *HP Vertica Essentials.* PACKT Publishing, Birmingham.

Balaskó Ákos – Lovas Róbert – Manolis, Nikos – Gergely Márk: Supporting Agricultural Communities with Workflows on Heterogeneous Computing Resources. *6th International Workshop on Science Gateways – IWSG 2014,* IEEE, Dublin, 18–23.

De Prato, Giuditta – Nepelski, Daniel – Piroli, Giuseppe – O'neill, Eoghan (2015): *Innovation Radar: Identifying Innovations and Innovators with High Potential in ICT FP7, CIP & H2020 Projects.* Publications Office of the European Union, Luxembourg.

Eifrem, Emil – Robinson, Ian – Webber, Jim (20152): *Graph Databases.* O'Reilly Media, Sebastopol.

Farkas Bálint – Kovács Gábor– Király István – Turóczy Attila – Kőnig Tibor – Érsek Attila – Safranka Mátyás – Fülöp Dávid – Pellek Krisztián – Kiss Balázs (2013): *Windows Azure lépésről lépésre.* Jedlik Oktatási Stúdió, Budapest.

Hajnal Ákos – Márton István – Farkas Zoltán – Kacsuk Péter (2015): Remote storage management in science gateways via data bridging. *Concurrency and Computation: Practice and Experience,* Vol. 27, No. 16. 4398–4411.

Harrison, Guy (2016): *Next Generation Databases – NoSQL, NewSQL, and Big Data.* Apress, New York.

Hewitt, Eben (2010): *Cassandra – The Definitive Guide.* O'Reilly Media, Sebastopol.

High Level Expert Group on Scientific Data (2010): *Riding the wave. How Europe can gain from the rising tide of scientific data. European Union.*

HP Enterprise: Closing the gaps in natural language processing. Technical whitepaper. Source: www8.hp.com/h20195/V2/GetPDF.aspx/4AA6-4691ENW.pdf (2016. 08. 24.)

Kacsuk Péter (ed.) (2014): *Science Gateways for Distributed Computing Infrastructures.* Springer, New York.

Kecskeméti Gábor – Gergely Márk – Visegrádi Ádám – Németh Zsolt – Kovács József – Kacsuk Péter (2014): One Click Cloud Orchestrator: Bringing Complex Applications Effortlessly to the Clouds. *Euro-Par 2014: Parallel Processing Workshops,* Lopes, Luís et al. (eds.), *Lecture Notes in Computer Science,* Vol. 8806, Springer, Cham, 38–49.

Konwinski, Andy – Karau, Holden – Zaharia, Matei – Wendell, Patrick (2015): *Learning Spark.* O'Reilly Media, Sebastopol.

Ludäscher, Bertram – Altintas, Ilkay – Berkley, Chad – Higgins, Dan – Jaeger-Frank, Efrat – Jones, Matthew – Lee, Edward A. – Tao, Jing – Zhao, Yang (2006): Scientific Workflow Management and the Kepler System. Special Issue: Workflow in Grid Systems. *Concurrency and Computation: Practice & Experience,* Vol. 18, No. 10. 1039–1065.

MongoDB Inc. (2015): *MongoDB Architecture Guide.* Source: www.mongodb.com/collateral/mongodb-architecture-guide (2016. 08. 24.)

Moreno-Vozmediano, Rafael – Montero, Rubén S. – Llorente, Ignacio Martín (2012): IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures. *Computer,* Vol. 45, No. 12. 65–72.

Pesce, Valeria et al. (2011): The CIARD RING, an Infrastructure for Interoperability of Agricultural Research Information Services. *Agricultural Information Worldwide,* Vol. 4, No. 1. 48–53.

Radez, Dan (2015): *OpenStack Essentials.* PACKT Publishing, Birmingham.

Sosinsky, Barrie (2011): *Cloud Computing Bible.* Wiley Publishing, Indianapolis.

Vasiliev, Yuli (2007): *SOA and WS-BPEL.* PACKT Publishing, Birmingham.

Wettinger, Johannes – Behrendt, Michael – Binz, Tobias – Breitenbücher, Uwe – Breiter, Gerd – Leymann, Frank – Moser, Simon – Schwertle, Isabell – Spatzier, Thomas (2013): Integrating Configuration Management with Model-Driven Cloud Management Based on TOSCA. *Proceedings of the 3rd International Conference on Cloud Computing and Service Science,* SciTePress, Aachen, 437–446.

Internet sources

Agrodat.hu projekt, website: www.agrodat.hu (2016. 08. 24.)
EU FP7 CloudSME projekt, official website: http://cloud-sme.eu (2016. 08. 24.)
EU FP7 IDGF-SP projekt, official website: http://idgf-sp.eu (2016. 08. 24.)
EU FP7 SCI-BUS projekt, official website: http://sci-bus.eu (2016. 08. 24.)
EU FP7 SHIWA projekt, official website: http://shiwa-workflow.eu (2016. 08. 24.)
HPE Helion Documentation: https://docs.hpcloud.com (2016. 08. 24.)
MTA Cloud, official website: https://cloud.mta.hu (2016. 08. 24.)
Occopus projekt, official website: http://occopus.lpds.sztaki.hu (2016. 08. 24.)