Chapter

Building a Big Data Platform Using Software without Licence Costs

Vassil Vassilev, Viktor Sowinski-Mydlarz, Pawel Gasiorowski, Sorin Radu, Sabin Nakarmi, Martin Hristev, Reza Baghaeishiva and Tarun Bali

Abstract

This chapter presents the experience in developing and utilizing Big Data platforms using software without license costs, acquired while working on several projects at two research institutions – the Cyber Security Research Centre of London Metropolitan University in the United Kingdom and the GATE Institute of Sofia University in Bulgaria. Unlike the universal computational infrastructures available from large cloud service providers such as Amazon, Google, Microsoft and others, which provide only a wide range of universal tools, we implemented a more specialized solution for Big Data processing on a private cloud, tailored to the needs of academic institutions, public organizations and smaller enterprises which cannot afford high running costs, or do significant in-house development. Since most of the currently available commercial platforms for Big Data are based on open-source software, such a solution is fully compatible with enterprise solutions from leading vendors like Cloudera, HP, IBM, Oracle and others. Although such an approach may be considered less reliable due to the limited support, it also has many advantages, making it attractive for small institutions with limited budgets, research institutions working on innovative solutions and software houses developing new platforms and applications. It can be implemented entirely on the premises, avoiding cloud service costs and can be tailored to meet the specific needs of the organizations. At the same time, it retains the opportunity for scaling up and migrating the developed solutions as the situations evolve.

Keywords: big data, AI, data platform, private cloud, public domain

1. Introduction

According to the recent Gartner report on strategic technology trends [1]. Platform Engineering is one of today's top 10 trends influencing enterprise strategies. *Data Platforms* deal with the entire lifeline of digital data from the moment it is generated at the data source through the communication channels which collect it and transport it to the destinations where it is accumulated, transformed, stored and processed, all the way to the actual interpretation of the result at the destination [2]. Behind its continuing expansion are two orthogonal but interconnected factors of development in the digital age – the evolution of computing technologies and the vast amount of digital data available from different sources which are potentially useful for different purposes.

The evolution of data processing in the digital age went through four main paradigms, constantly vacillating between centralization and decentralization:

- Localization of the data processing in a single physical place (*desktop*, *embedded* or *mobile* device). Dominant during the early computing and communication devices.

- Distribution of the processing power across multiple physical locations on a computer or communication network (*client-server* or *peer-to-peer* architecture). Exploits the opportunities for sharing provided by contemporary digital networking on local, regional or global scale.

- Concentration of the processing power by creating virtual locations for hosting multiple processors (*cloud* or *edge* computing). Initially supported and later even enforced by the big vendors because it facilitates the performance of computations on a scale previously not imaginable and unaffordable.

- Virtualization of the computation by creating logical processors in different physical and virtual locations (*blockchains* or *data spaces*).

Since the beginning of the new Millenium this conceptual evolution was accelerated by the evolution of software technologies. Firstly, two complementing enabling technologies contributed to this evolution: *virtualization* of the computational infrastructure and *containerization* of the execution environment (see **Figure 1**). While the virtualization allows applications to be executed in the environment of a virtual operating system, chosen by convenience, containerization allows the execution to be identical, regardless of the deployment location. The contemporary cloud provision for deployment employs both approaches, thus supporting the high scalability of the hardware and software infrastructure as well as the mobility of the applications across the platform locations. The transparency of the application deployment in such a case is provided by the container management system, which is a key element of each cloud provision [3].

This shift of the computational paradigms is paired with the expansion of the software systems towards service-oriented architectures (SOA), where the applications are not executed in isolation but as an element of a *service workflow* (see **Figure 2**). The platforms with SOA architecture can *orchestrate* the services, thus adding the possibility for automation of workflow planning and execution.

Unlike cloud computing, which essentially pushes for centralization of data processing on the cloud, the SOA better supports distributed architectures for peer-to-peer data processing in blockchains and data spaces. However, the two architectures are not antagonistic, since the service orchestration is also popular on the data platforms, while they may also act as data service providers or data service consumers in data spaces [4].

What is extremely encouraging is that all these opportunities are freely available and can be implemented in-house, tailored to the needs of the organization. This chapter presents the experience in implementing data platforms on a private cloud for

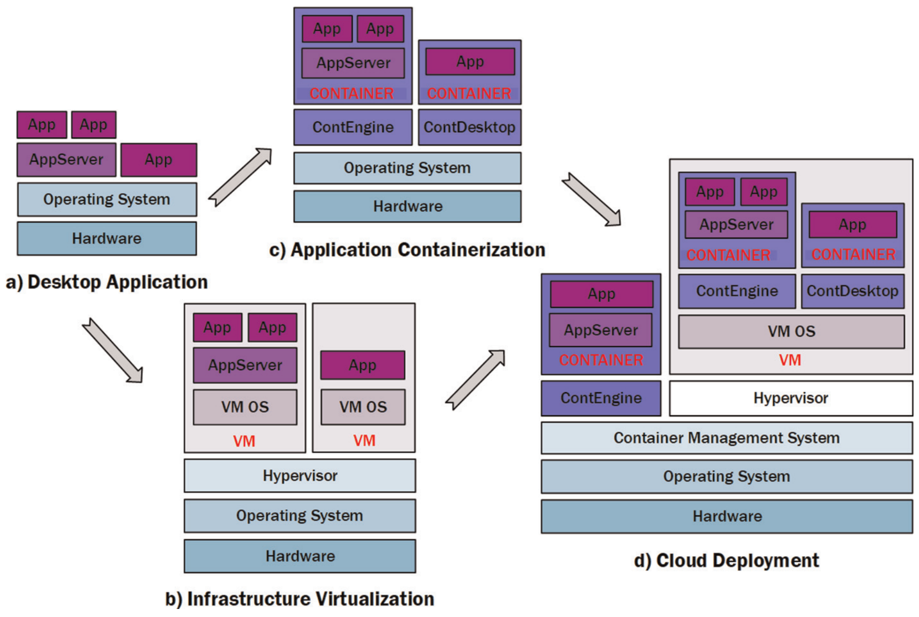


Figure 1.
Evolution of data processing from the desktop to the cloud.

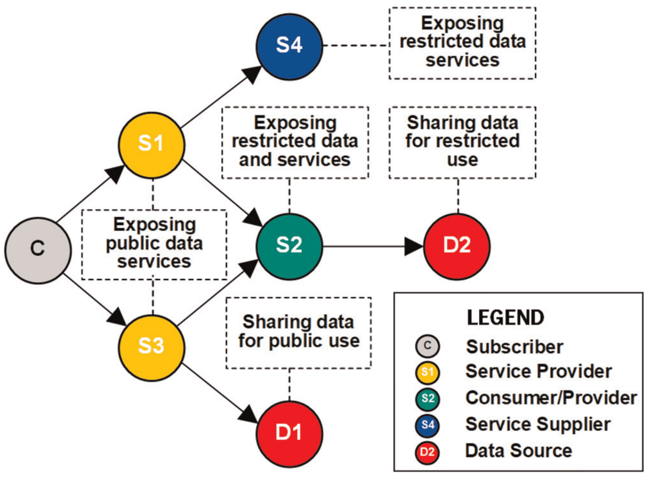

C
S1
S2
S3
S4
D1
D2
Exposing restricted data services
Exposing restricted data and services
Sharing data for restricted use
Exposing public data services
Sharing data for public use
LEGEND
Subscriber
Service Provider
Consumer/Provider
Service Supplier
Data Source

Figure 2.
Data processing workflow in a service-oriented architecture.

processing Big Data using software without license costs, which can be obtained from the public domain, or as community editions of commercial products.

The plan of the chapter is as follows. In the next section, following a brief review of some of the existing solutions, we will discuss the main alternatives which drive the design of data platforms. In the subsequent section, we will describe the main components of the platform. After that, we will present several pilot projects, which we implemented using such platforms – one for real-time security analytics, one for outdoor air quality monitoring and one for more complex urban development combining both outdoor and indoor environment factor analysis. At the end we will discuss the lessons learned, will formulate some recommendations and will discuss some future enhancements of the platform.

2. Alternatives and choices: models, technologies and tools

Data platforms and, specifically, Big Data platforms are complex systems which combine powerful hardware and software. They target decision-making, business organization and operation management. On the one hand, the main players in the market for enterprise software offer their own tools and whole ecosystems for data processing and Big Data management. Both traditional software powerhouses, such as IBM, Hewlett-Packard, Oracle and Amazon in the United States and SAP in Europe [5–9], as well as some of the purpose-built software companies specialized in marketing Big Data tools, such as Cloudera [10] are offering enterprise suites with extremely powerful data management and data analysis capabilities, based on well-established concepts originating in the open-source community [11]. On the other hand, the global service providers, such as Amazon, Google, and Microsoft [12–14] are hosting most of these tools on their own cloud premises thanks to the technologies of virtualization, containerization and orchestration which are the foundation of cloud computing. In some cases, this symbiosis goes even further by embedding mechanisms for utilization of the specific storage infrastructure, like the recent Cloudera platform **CDH**, which is seamlessly integrated with the object storage of **AWS** [12]. However, although midsize companies and smaller software houses prefer to rely on software vendors and service suppliers, the price tag associated with it is exorbitantly high. In most cases, it is out of reach for many private and public organizations, and they typically look for a bespoke solution on their own premises instead. An additional advantage of having custom-built data platforms on the premises is the compatibility and the opportunity for subsequent migration to an enterprise platform. Since most of the current data platforms are assembled out of software products and systems which originate in the open-source community, in the case of scaling up these in-house platforms, it would be possible to migrate the applications much more easily. This section will systematically consider the necessary decisions to prepare ourselves for building a custom-tailored in-house platform for processing Big Data on a private cloud. This would be a competitive solution for data-intensive but relatively small organizations such as research institutes, local councils, public agencies and SMEs.

2.1 Conceptualization of data processing, workflows and data services

To build a platform which supports the entire lifecycle of the data from the moment it has been generated until the moment the result of its processing is interpreted, we need to have a clear conceptual understanding of the use of such a platform for its intended use. The most economical and useful way to do it is to have a model, presented graphically using a set of diagrams, which is complemented by a description of the typical business scenarios and detailed technical specifications:

- Data: data sources, data formats and data pre- and post-processors.

- Platform infrastructure: hardware and software components and subsystems.

- Data processing architecture: software pipelines and data flows.
- System interaction: communication ports, protocols and operations.
- Operation timelines: scenarios for using the platform for specific business tasks.

From a software engineering perspective, the standard way of doing this is using UML as a modeling language. It provides the most complete set of diagrams, which makes the design informative and technically sound, but it may also lead to "analysis paralysis" in complex tasks such as the platform design. Because of this, we prefer a logical approach, which combines the modeling of the conceptual ontologies with flowcharts of their use. It also guarantees consistency across the diagrams but is more economical than UML. As an example, **Figure 3** shows the ontology of data processing. Some of the concepts in it are physical and will later become software components of the platform, while others are purely logical, helping to understand the problem, conceptualize the processes and explain the results of processing the data on the platform. In ontology, we can model the data processing operations as relations between the states and organize them into a hierarchy, similar to the concept taxonomy. The full ontology of the platform, as we modeled it, includes more than 600 different concepts, but the development of the platform can be incremental so that the ontology can be a useful starting point for the design. As a step towards this, **Figure 4** illustrates the sharing of both data and services for processing it across several business projects analyzing the environment impact on urban life. Three projects are considered "first level": monitoring outdoor air pollution (Project 3), analyzing the medication prescriptions in the local pharmacies (Project 2) and mapping them in the local

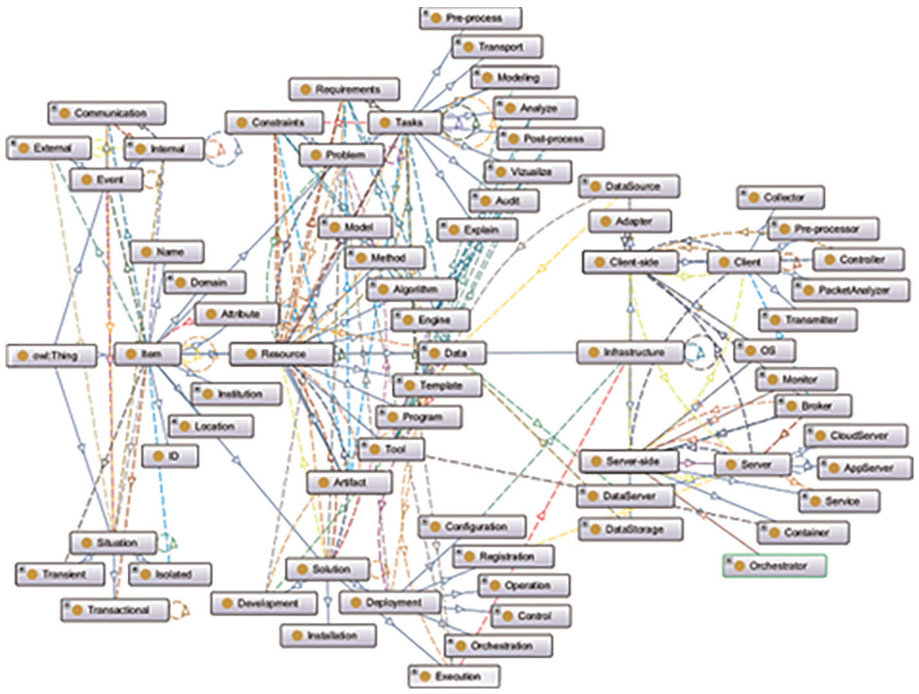


Figure 3.
Ontology of data processing on the data platform (core fragment).

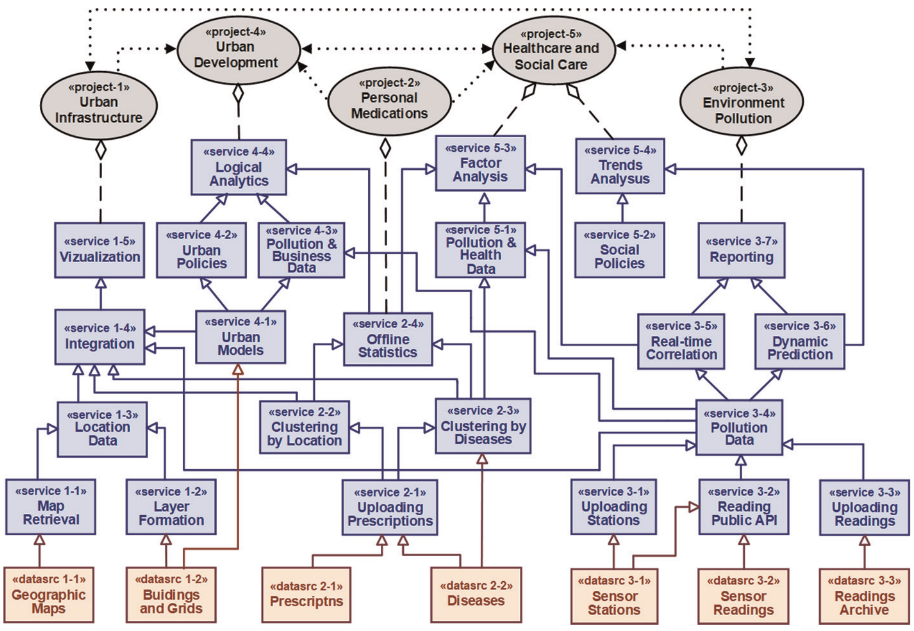

«project-4» Urban Development
«project-5» Healthcare and Social Care
«project-1» Urban Infrastructure
«project-2» Personal Medications
«project-3» Environment Pollution
«service 4-4» Logical Analytics
«service 5-3» Factor Analysis
«service 5-4» Trends Analysis
«service 1-5» Vizualization
«service 4-2» Urban Policies
«service 4-3» Pollution & Business Data
«service 5-1» Pollution & Health Data
«service 5-2» Social Policies
«service 3-7» Reporting
«service 1-4» Integration
«service 4-1» Urban Models
«service 2-4» Offline Statistics
«service 3-5» Real-time Correlation
«service 3-6» Dynamic Prediction
«service 1-3» Location Data
«service 2-2» Clustering by Location
«service 2-3» Clustering by Diseases
«service 3-4» Pollution Data
«service 1-1» Map Retrieval
«service 1-2» Layer Formation
«service 2-1» Uploading Prescriptions
«service 3-1» Uploading Stations
«service 3-2» Reading Public API
«service 3-3» Uploading Readings
«datasrc 1-1» Geographic Maps
«datasrc 1-2» Buidings and Grids
«datasrc 2-1» Prescriptns
«datasrc 2-2» Diseases
«datasrc 3-1» Sensor Stations
«datasrc 3-2» Sensor Readings
«datasrc 3-3» Readings Archive

Figure 4.
Data platform as a service-oriented architecture (urban life projects).

area using public sources of geolocation information (Project 1). Additionally, there are two cross-domain projects for secondary analysis of the impact of environmental pollution on public healthcare (Project 5) and on urban development (Project 4), which use data, existing services, and some of the results produced by the "first-level" projects. They can provide deeper, policy-driving analysis of the environmental impact based on factors, trends and heuristic methods. The diagram shows the data sources, data flows and dependencies between the services. It is sufficient for the design of software architecture according to both the privacy and the sharing policies.

The conceptual model must lead to a practical implementation of the scenarios of use and the most natural way to do this is to map it to a service-oriented architecture (SOA). All aspects of the SOA – functional, temporal, causal, etc., can be designed so that they retain semantic consistency with the conceptual model. The integrity can be guaranteed by stereotyping the ontological entities, which preserves the meaning across different diagrams, similar to the stereotyping in UML. For our example, **Figure 5** shows the workflow of correlating data in the form of a directed graph, which implements the "Real-time Correlation" service from **Figure 4**.

We have found this approach more useful than the use of UML at the early stage of design of the platform since it ignores low-level details and combines many different aspects of the model in a single diagram, which facilitates early grasping of the essentials and organizing the work on a technical level in an optimal way. The next sections will go systematically through the alternative choices and detailed descriptions of the tools which map this conceptual model to a working system.

2.2 Exploring data and metadata

The enterprise data platforms, as well as the platforms built on the public cloud, have configurable components from a chosen software *ecosystem*. Their universality is

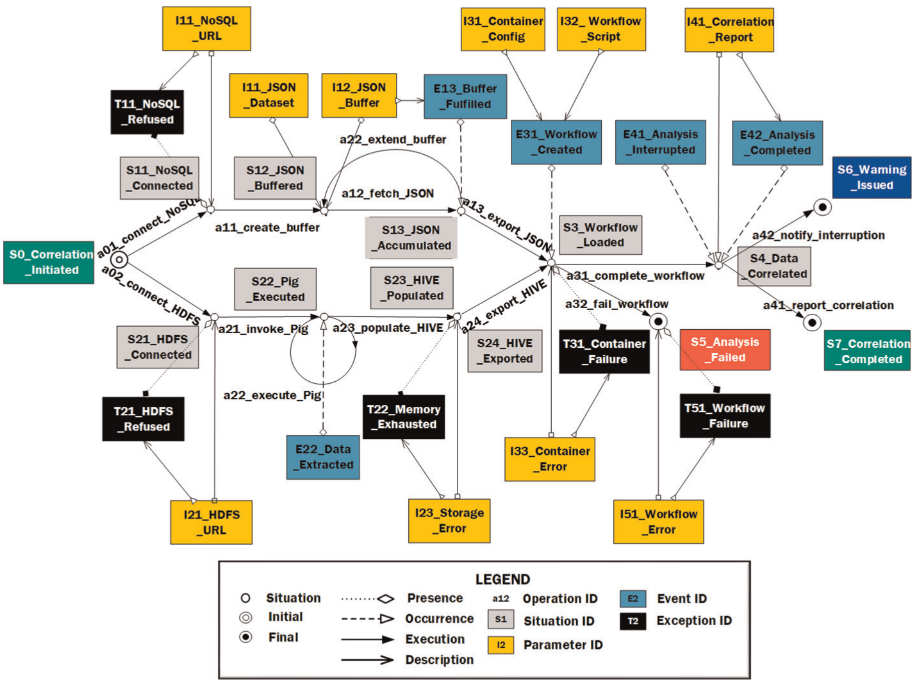


Figure 5.
Workflow for data processing on the platform (data correlation scenario).

restricted only by the software vendor, or by the support provided by the public cloud to the installed ecosystem. The private cloud-based platforms do not have these limitations, although their scope is more limited. This allows custom-tailored design, which better accounts for the specific data to be processed on the platform. Quite a few characteristics of the data need to be considered at this stage:

- The data differs significantly in a variety of ways – formats, granularity, volume, noise, location, etc.
- Data processing is performed along a complex workflow of operation – sampling, aggregation, buffering, feature selection, training, validation, analyzing, merging, interpretation, explanation, etc.
- The tasks for data analysis have large diversity – detection, recognition, classification, correlation, factorization, prediction, etc.
- For each analytical task there is a whole variety of methods with different applicability – temporal, structural, logical, model-driven, behavioral, hybrid, etc.
- To reach wider community of users the data processing needs to be comprehensive by providing statistics, reports, explanation, etc.

This variety directly affects the subsequent choices of technologies, methods and tools for building the ecosystem of the platform. The main technical characteristics of

Data types	Sources	Content	Ingestion	Transport
Samples (structured data)	networks, hardware and software	readings, packets, locations	one-off	memory sharing, parameter passing
Artefacts (unstructured data)	media editors, reporters and cameras	documents, images, videos	one-off	FTP/S, HTTP/S, SSH, FTAM, WebDAV, WebSockets
Messages (semistructured data)	messengers, devices and monitors	alerts, logs, messages, emails	one-off	MQTT, AMQP, SMS, IRC, XMPP, ModBus, Websocket, RCS
Streams (fully/ semi-structured sequences)	signal emitters, trackers and video cameras	timeseries, broadcasts, feeds	continuous	HLS, WebRTC, RTSP, RTMP, SRT, MPEGDASH, ModBus
Datasets (fully/ semi/unstructured collections)	spreadsheets, databases and simulators	descriptions, operations, locations, etc.	one-off, batch	FTP/S, HTTP/S, AFTP, OFTP, AS2, WebDAV
Repositories (collections of datasets)	databases, warehouses and data lakes	mixed	batch	supported by the repository

Table 1.
Data characteristics affecting the platform design.

the data which may have an impact on the design choices are shown in **Table 1**. Particular attention is needed for the processing of time series in real-time [15].

Creating data models is one of the early tasks which can be addressed during the design of the platform since it helps both the processing and the persistence of the data. Depending on the data sources, different approaches can be used for modeling: purely relational using ER diagrams, object-relational using UML class diagrams, hypertext using XML or JSON and logical using semantic languages like RDF/RDFS/OWL. An additional important factor to help understanding the data and preparing it for subsequent processing is the *metadata,* the information about the data. It can be used for data cleaning and filtering, for semantic enrichment by adding missing attributes and establishing missing links, as well as for preparing persistent storage and efficient retrieval of the data via semantic indexing.

2.3 Choosing methods for data processing

Data processing on the platform may occur in a variety of contexts and can serve different purposes, so the selection of suitable methods for each task will affect the choice of tools during the development phase:

- Different stages of data pipeline: at the source, before transmission, during transmission, on arrival, before storing, inside the repository, etc.

- Different structure and format of the data: structured (CSV, SQL), semi-structured (JSON, XML, RDF, SVG, etc.) and unstructured (binary, document, graphics, etc.)

- Different preparation of the rough data: filtering, formatting, anonymization, normalization, enrichment, aggregation, buffering, accumulation, etc.

- Different tasks of the processing: detection, classification, recognition, correlation, profiling, prediction, etc.

- Different methods behind the algorithms: statistical, clustering, graph-based, rule-based, model-based, optimization-based, etc.

- Different interpretations of the results: simple reporting, black box explanation, white box explanation, impact factor analysis, etc.

Constructing an enterprise-quality data platform requires well-proven methods for which there are mature tools, guaranteeing robust and reliable operation. In some cases, the data processing may be implemented using publicly available libraries; in other cases, there are no suitable tools, and bespoke software needs to be developed with *Python* as the ultimate language of choice for programming. Although it might be assumed that the more sophisticated methods for data analysis are not easy, with the huge advancement in statistical, behavioural and machine learning methods, their use is a completely feasible task supported with a huge amount of software libraries. Less advanced methods are a bigger problem, since there is rarely a universal and high-quality software available to implement them.

2.4 Software tools for data management and data processing

The technology stack of the platform includes software components for processing the data along its lifecycle, from the sources to the presentation of the results. For most of the necessary tasks, there are software products in the public domain, and many enterprise software products also have community editions, so the composition of a relatively universal data platform using software without license fees is absolutely feasible. **Table 2** contains some software sufficient to compose a technology stack powerful enough for a wide class of typical Big Data scenarios. We used most of them

Operations	Description	Software products
Data Preprocessing	Processing at the source, in transition, before analysis	**Hackolade** (modelling), **Annotator** (annotating), **Amnesia** (anonymizing)
Data ingestion	Transporting files, exports, messages, streams,	**Mosquito** (messages), **Kafka** (streams), **NiFi** (files), **Hop** (general)
Data persistence	Storage in structured, hypertext and binary format	**PostgreSQL** (SQL), **MongoDB** (JSON) **Neo4J** (RDF), **3DCityDB** (CityGML)
Data Postprocessing	Transformation, analysis and integration	**Python** (data-), **Java** (operation-), **JS** (Web-) and shell scripts (OS-centric)
Big data management	Storing, retrieving, mapping & searching Big Data	**Hadoop** (pairs), **Storm** (tuples), **Cassandra** (tables), **HPCC** (clusters)
Big data analysis	Analyzing Big Data using statistical, ML & RL	**Spark** (analysis of distributed data), **Storm** (analysis of streaming data)
Interpretation	Data integration, reporting and visualization	**JS** (texts), **Jupyther** (diagrams), **Cesium** (maps), **Grafana** (general)

Table 2.
Technology stack for data processing on big data platforms.

in several projects, and as we will show in the next section, it is both powerful and highly scalable for working on multiple projects.

A separate consideration is needed for the choice of components for Big Data processing. Our platform is based on Apache Hadoop, which has one of the oldest Big Data ecosystems [16]. It maintains the data in files, which makes it inherently slower. There are some more recent and modern alternatives in the public domain, such as Cassandra, Storm, and HPCC, plus some highly scalable no-SQL databases like Redis, CouchDB and OrientDB [17], which might be more suitable for specific characteristics of the Big Data, like speed of growth, degree of dependency, etc. Our choice of Hadoop was dictated by the need for compatibility with our academic and business partners. Being an originator of many enterprise systems, it may be seen as an "old-fashioned" choice, but Hadoop has a rich ecosystem which is compatible with many commercial systems, so it is still attractive.

The use of additional databases besides the main Big Data repository may seem redundant at first sight but it is justified. Firstly, the rough data, the pre-processed data and the archived data can be used in different projects for different purposes. Keeping the data in a single place in a single format may lead to inefficiency due to the need for conversion. Secondly, sometimes, it is preferable to store specific data in its original format for specific operations, supported natively by specialized database systems. This applies to all non-SQL data – hypertext in JSON format (MongoDB), graphs in RDF format (Neo4J) and 2D/3D in CityGML format (**3DCityDB**).

The separation of preliminary data processing from Big Data processing leads to two different approaches for supporting the analysis of the data. In the Big Data cluster, this can be done using the tools from the Hadoop ecosystem (we use Apache Spark [18]), while in the temporary area of holding the data, it can be processed within the respective databases using their own APIs or externally, using Python.

3. Building the data platform

Our motivation for developing a private cloud-based data platform comes from our interest in the automation of data processing using explicit policy rules for controlling the execution. Initially, we incorporated some of these ideas in the data processing framework for threat intelligence and security analytics of the Cyber Security Research Centre [19]. In another project, we needed computational resources on the public cloud and hosted the software components on Amazon AWS [20]. This experience led us to the concept of implementing a data platform on our own private cloud, which resulted in the solution reported here [21]. In this section, we will describe the main steps of the process.

3.1 Hardware and system software

The starting point for the implementation of the data platform for processing Big Data on a private cloud is the hardware infrastructure. In a realistic scenario, cloud technology requires multiple hosts organized in a virtual cloud infrastructure according to the principle "the more memory – the better". However, being academic institutions primarily focused on research and innovation, both in London and in Sofia, we started with a single server host first. Currently, we are running our platforms on hosts with 128GB RAM/256GB RAM, equipped with 80 TB/120 TB hard disk space (in London/Sofia, respectively). Such capacity might be considered a minimum

Platform role	Description	Software products
Platform management	Register, allocate and control resources	**Nginx** (access control), **Proxmox** (admin, develop, deploy)
Virtual machine management	Emulate different OS on the infrastructure	**Linux KVM** (virtualize)
Container management	Distribute, isolate and synchronize components	**Docker** (containerize), **Kubernetes** (locate), **Zookeeper** (synchronize)
Workflow management	Compose, schedule and integrate workflows	**AirFlow** (orchestrate), **Flask** (integrate)
Process management	Monitor and report the execution	**MLFlow** (monitor, report)
Service management	Search, retrieve and report the consumption	**Elasticsearch** (search, retrieve), **Kibana** (accumulate, report)

Table 3.
System tools for managing of platform resources.

requirement to provide an adequate environment for several teams to work on different projects. Thanks to the scalability of the platform, at a subsequent stage, it can be migrated to a more capable hardware infrastructure with multiple hosts and much larger memory space, as necessary.

Secondly comes the system software, which provides support for the development, deployment and execution of the software components of the platform. This system software is completely independent of the data which the platform is going to process and is neutral to the possible applications deployed to it. The system software which we used for this purpose is shown in **Table 3**.

All system software can be used without license costs and can be obtained from the respective public sites on the Internet. The community versions of enterprise products may have some limitations, but they are still fully functional and can be used for the purpose in non-commercial environment.

3.2 Assembling the platform

Endorsing the previous recommendations leads to the possibility of building an entirely user-empowered platform for processing Big Data on a private cloud, fully compatible with more sophisticated enterprise platforms from the big vendors [5–10]. The software architecture of a similar solution, like our data platform, is shown in **Figure 6**. It is formed out of five different groups of software components:

- Platform management (platform-specific but service and data independent) – management of the platform resources: Kubernetes for container management, Docker for containerization, Proxmox for remote access and resource management, Nginx for security control, Elasticsearch and Kibana for auditing and reporting, Airflow and MLFlow for workflow and process scheduling and monitoring.

- Data ingestion (service specific but data independent) – Mosquitto, Kafka, Nifi and the general-purpose Hop for managing the communication channels, transportation, ingestion and storing.

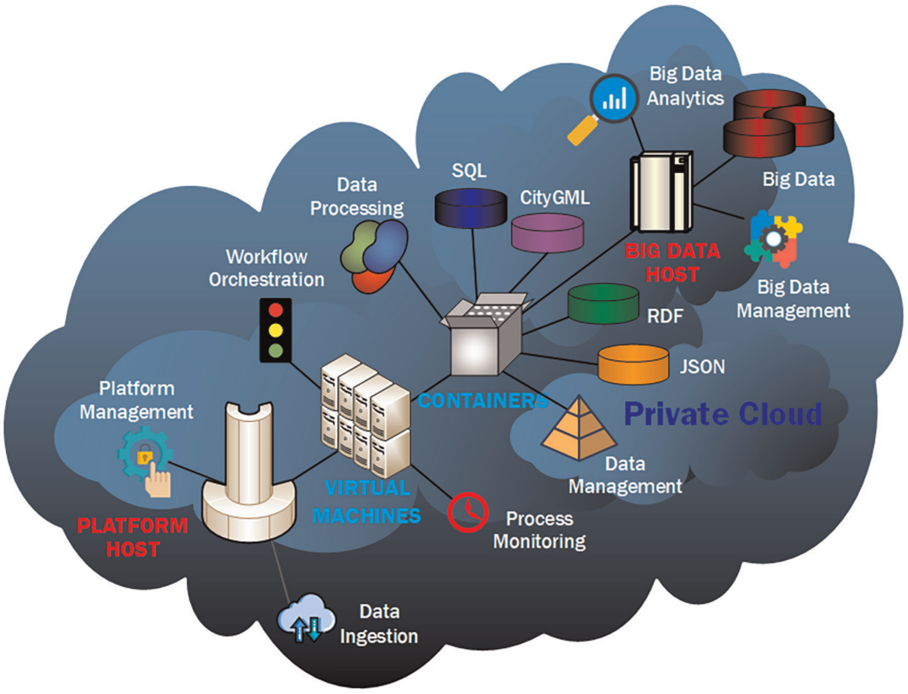


Figure 6.
System architecture of a cloud-based data platform (two hosts version).

- Data repositories (data specific but service independent) – database management systems for temporary data storage of data in four different formats – SQL (PostgreSQL), JSON (MongoDB), RDF (Neo4J) and CityGML (**3DCityDB**).

- Data management and data processing (data and service-specific) – bespoke software components developed to meet the specific requirements of the applications for managing and processing the data along its entire lifeline.

- Big data cluster – it incorporates the ecosystem for Big Data management and analysis. In our case, we used tools from Hadoop ecosystem, but they can be substituted or complemented with tools from other ecosystems. The use of Spark or Storm for analysis is not restrictive, either, since Hadoop can run directly Python and Java.

All software components are based on software without license costs from the public domain and community edition of enterprise products, selected to provide support for development, deployment and operation as discussed earlier. Their choices can be easily adapted to meet the specific requirements of the organizations.

Although the proposed architecture cannot match the universality, scalability and extendibility of public cloud provisioning, it works sufficiently well for many customers due to its simplicity and flexibility. The separation of platform management from data management, and the temporary data from the Big Data leads to an open architecture, highly scalable and extendable on both platform and layer levels. At the Cyber Security Research Centre of London Metropolitan University, the Big Data cluster is installed on a single host and accommodates only 80 TB of data, while at GATE Institute of Sofia University, it spans many physical hosts with a total memory capacity measured in petabytes. At the same time, at the Cyber Security Research Centre, all other components of the platform are installed on a single physical host, while at GATE Institute the separate groups of components are installed on different hosts to support working on multiple projects in parallel and to handle much larger amounts of data.

3.3 Setting up the deployment context of software components

Unlike ordinary information systems, which operate isolated from other software systems, the applications running on the platform consist of components which are dependent on each other within their context of execution. The data in different applications may come from the same data sources, while different data processing pipelines may use the services provided by the same components. This leads to the need for putting suitable control mechanisms in place for communication *tracking*, data *access control*, component *isolation*, process *synchronization and* workflow *orchestration*. The virtualization and containerization mechanisms on the cloud support a variety of options for achieving this. **Table 4** presents the alternative deployment contexts available for both the design of platform components used in all applications, as well as for the design of specific application components in different projects.

In SOA, each software component can be seen as a server, accessible through a dedicated TCP/IP port and exposing services on a different level – administration, implementation, and operation and at different stages of working – designing, developing, deploying and using. From this perspective, the TCP/IP ports of the engines executing the component services can be opened using three methods:

- Static IP address on the Internet, if needed, to be accessible by the physical users.
- Dynamic IP address, valid locally only and accessed by the different tools installed on the operating systems of the host computers or the VMs running on them.
- Dynamic IP address accessed programmatically from within the permitted VMs or containers of the platform according to the deployment of the software component.

The choice of a suitable context for component deployment can be made at design time, but since it may significantly affect the development, it must be made carefully. There are several design considerations to be accounted for, the most important being the degree of sharing and isolation of the services. Installing the components directly under the control of the operating system (*"bare metal")* provides the widest possibility for sharing, but at the same time, it guarantees only very low isolation, while containerizing them provides the highest level of isolation but lowers the possibility for synchronization. An additional consideration which may need to be accounted for during the design is the visibility of the communications and the need to protect the information privacy and operation security. The closer to the operating system, the less control mechanisms can be used, while encapsulating the components within containers provides multiple mechanisms for controlling them.

Engine	Function	Context	Software support
Operating system (OS)	Resource allocation	Hardware	**Linux**, **Windows**, **MacOS**
Hypervisor (HV)	Resource virtualization	OS	Linux **KVM**, VMWare **vSphere**, MS **Hyper-V**, Oracle **VBox**
Virtual machine (VM)	Resource isolation	OS or HV	Ubuntu **Linux**, MS **Windows**, Oracle **Solaris**, Apple **MacOS**
Container manager (CM)	Component virtualization	OS or VM	K8 **Kubernetes**, Apache **Mesos**, HashiCorp **Nomad**
Container (CNT)	Component isolation	OS, VM or CM	**Docker**, **LXC**, **Windows Containers**, **Podman**
Server (SRV)	Service isolation	OS, VM or CNT	**NodeJS**, **JupytherHub**, **MongoDB**, **GlassFish**, **PostgreSQL**, **Neo4J**
Runtime (RT)	Service execution	OS, VM, CNT or SRV	Language-specific (programme interpreters, script shells, etc.)
Component	Service	OS, VM, RT CNT or SRV	Task-specific (data management, data analysis, visualization, etc.)

Table 4.
Context of deployment of software components on the cloud.

3.4 Enforcing security policies

The last step is to open the TCP ports for access to the services. Since each component exposes a service, the access to platform services from external clients or internal components is through a corresponding port. This can be done using two methods – control of the port visibility in accordance with the security policies (see **Table 4**) and allocating operational rights to external users according to their profiles. The first requires service registration after component deployment and must be updated for each new application, while the second requires user profiling.

4. Pilot projects

In this section, we will present three pilot projects, which have been completed using the platform by a joint team of London Metropolitan University and GATE Institute in three different projects – Computer network security analysis [22], Environment pollution monitoring in Sofia [23], and Impact of environment pollution on public health in London [24]. The successful completion of the projects proves the viability of the concept and its potential for use by public organizations, NGOs, SMEs and academic institutions dealing with Big Data.

4.1 Computer network traffic analysis (real-time pilot)

This project was the first pilot test of the data platform implemented after the concept described previously. It was dedicated to the detection of unsolicited behaviour due to unauthorized intrusion and/or the presence of malicious software. The goal was to analyze the network traffic and the event logs in real time to detect potentially missed interventions, as well as to perform a secondary analysis by examining the network traffic over a longer period. As a data source, we used a staged environment, running applications with malicious software attached to them. The generated data was captured by network analysers and transported for further analysis on the cloud over two protocols – TCP for the network packets and MQTT for the logs generated within a staged environment.

We performed two different analytics – real-time pre-processing and Pearson correlation of the streams and packet classification and intervention recognition using machine learning algorithms. The correlation analysis did not produce very exciting results due to the insufficient data generated by the simulator and the different speeds of the streams, but the subsequent forensic analysis using standard machine learning algorithms (logistic regression, neural networks and SVM) and deep learning (we used seven-layer CNN) was very interesting. It showed that although deep learning definitively produces the best results, it might not be the most suitable since some of the classical machine learning algorithms, which are much simpler, produce almost as good results. In our case, the SVG algorithm produced results close to the results obtained using CNN (see **Table 5**).

In this pilot, we used Docker for containerization of MongoDB as data storage and several data management components – for transportation (Mosquitto and Kafka),

Method	No. packets	RST (%)	ACK (%)	SYN (%)	Avg (%)
Regression	6268	85	81	31	66
SVM	6268	94	84	96	91
NN	6268	88	72	94	85
CNN	45,000	90	91	92	91

Table 5.
Precision of different methods for prediction of network packets (in %).

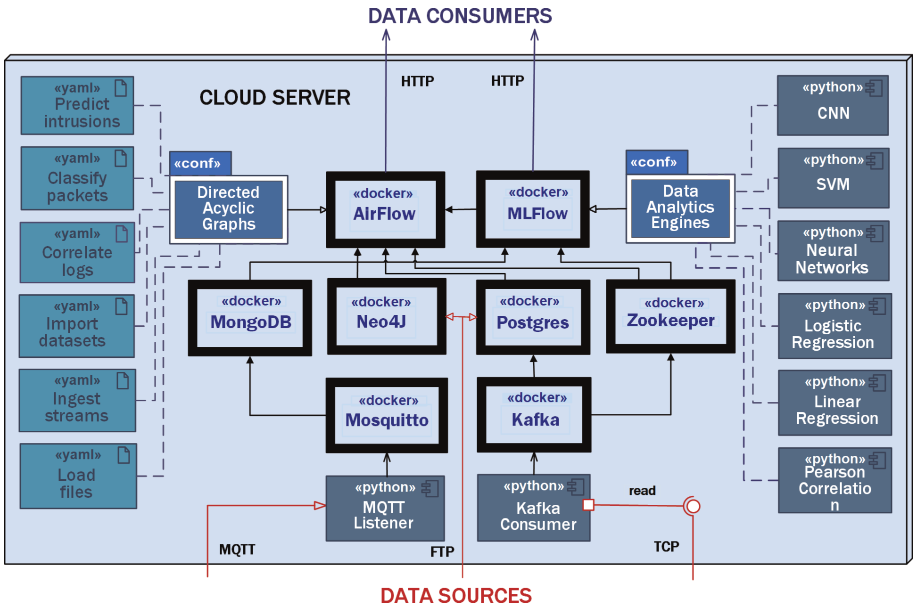


Figure 7.
Software components of the cloud-based data platform (portal host only).

operation synchronizing (Zookeeper), integration (Flask), workflow orchestration (AirFlow) and monitoring (MLFlow) – see **Figure 7**. Since we used only one physical host, the container management system did not play a significant role in the project.

The project was the first valuable test of the platform thanks to the data processing in both motion and peace, as well as the variety of methods for processing data both online and offline [22]. It is also interesting that due to the lockdown, the way in which the team of Cyber Security Research Centre of London Metropolitan University worked changed drastically – the staged environment was created earlier in London, while the actual platform for data processing is on the private cloud server of GATE Institute in Sofia, where the data was sent over the Internet. Both the development and the deployment of bespoke software were done remotely from London, which proves additionally the viability of the whole concept.

4.2 Environment pollution monitoring (Sofia pilot)

This project was focused on the environmental monitoring in Sofia due to the significant air pollution in the city [23]. The air pollution data was collected from 13 sensor stations across the city. The readings were formatted as JSON objects and sent to the cloud over MQTT for analysis. Before the data was accumulated in the MongoDB database, it was correlated using the standard Pearson algorithm for establishing dependence between different factors of pollution like temperature, humidity, gases, and particles in the air (see **Table 6**). The archived data from previous periods was uploaded to the same database using NiFi to train the machine learning algorithms before prediction.

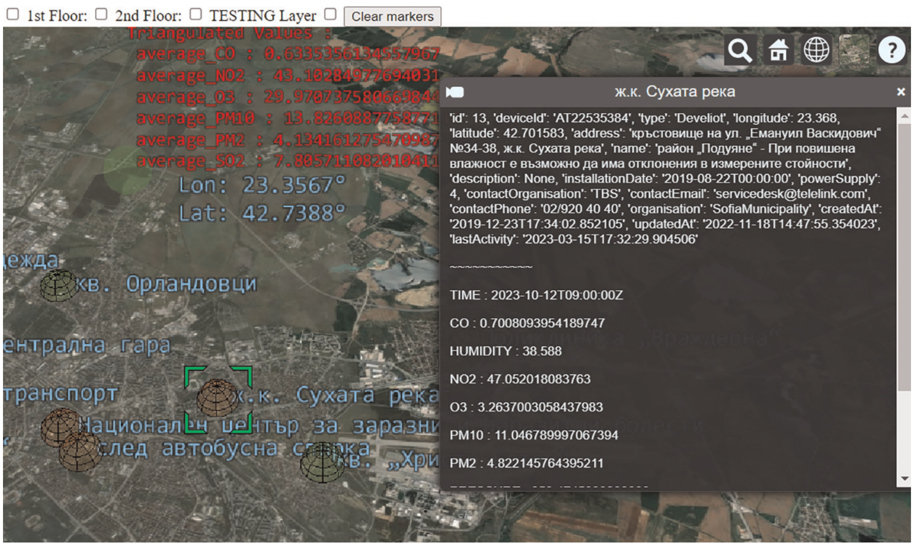
An essential addition to the platform in this project was the integration of the Cesium GS Cesium JS component, which allows multi-layered visualization on top of 2D maps [25]. As a 2D base, we used the free service provided by the OpenStreetMap Foundation [26], which has worldwide coverage. We integrated and visualized a variety of data from different data sources using two methods:

- Pop-ups: By combining pop-up windows rendering data from different sources with the visual stream of the map. The pop-up window on **Figure 8**, for example, combines sensor data from MongoDB with ontological information from OpenStgreetMap. We have also experimented with adding an explicit ontological model of the urban area in RDF format extracted from the Neo4J graph database [24].

- Projections: By superimposing additional layers on top of the base 2D maps to add urban infrastructure and 3D building models. In **Figure 9**, for example, the 3D

No	CO	SO_2	NO_2	O_3	PM2	PM10	Press	Hum	Temp
1	0.5039	6.3676	15.730	43.096	5.8484	8.291	955.855	46.322	16.202
2	0.4969	5.4908	38.971	53.154	8.6820	15.797	933.432	46.311	16.578
3	0.5010	4.7727	11.779	47.642	3.5257	5.965	959.594	96.717	20.416
4	1.1032	8.7043	9.9153	37.223	7.3437	15.251	960.677	31.885	18.247
5	0.3916	6.8086	14.5090	35.469	6.9110	15.995	959.174	54.637	12.476

Table 6.
Positive correlation between outdoor temperature and humidity (Pearson 0.972).



```
Triangulated Values :
average_CO : 0.6335356134557967
average_NO2 : 43.10284977694031
average_O3 : 29.97073758066984
average_PM10 : 13.8260887758771
average_PM2 : 4.134161275470987
average_SO2 : 7.805711082010411
Lon: 23.3567°
Lat: 42.7388°
```

ж.к. Сухата река

```
'id': 13, 'deviceId': 'AT22535384', 'type': 'Develiot', 'longitude': 23.368, 'latitude': 42.701583, 'address': 'кръстовище на ул. „Емануил Васкидович“ №34-38, ж.к. Сухата река', 'name': 'район „Подуяне“ - При повишена влажност е възможно да има отклонения в измерените стойности', 'description': None, 'installationDate': '2019-08-22T00:00:00', 'powerSupply': 4, 'contactOrganisation': 'TBS', 'contactEmail': 'servicedesk@telelink.com', 'contactPhone': '02/920 40 40', 'organisation': 'SofiaMunicipality', 'createdAt': '2019-12-23T17:34:02.852105', 'updatedAt': '2022-11-18T14:47:55.354023', 'lastActivity': '2023-03-15T17:32:29.904506'

~~~~~~~~~~~~

TIME : 2023-10-12T09:00:00Z

CO : 0.7008093954189747

HUMIDITY : 38.588

NO2 : 47.052018083763

O3 : 3.2637003058437983

PM10 : 11.046789997067394

PM2 : 4.822145764395211
```

Figure 8.
Data integration and visualization of the air pollution in Sofia.

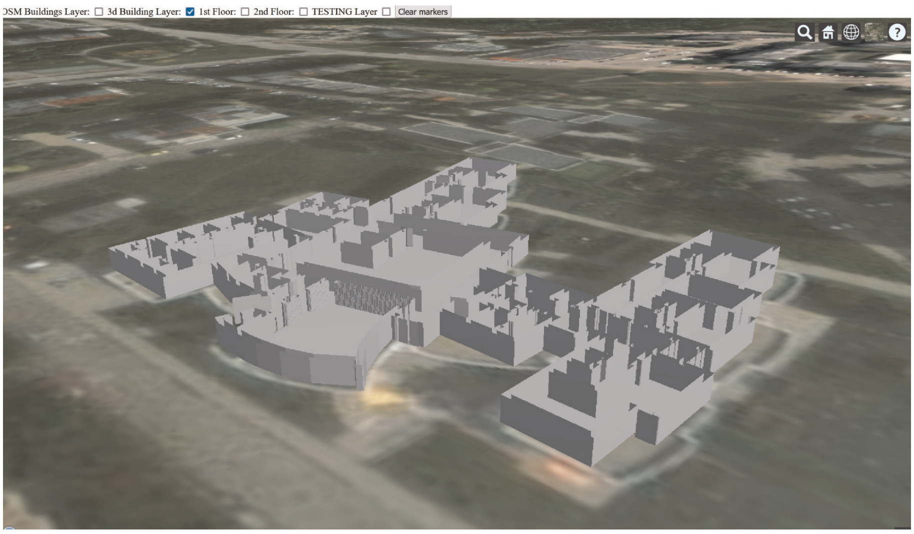


Figure 9.
Integrating 3D buildings model and ontological information with 2D map.

model of the building which has been reconstructed offline from its 2D floor plan, is subsequently embedded in the map to show the location of indoor sensors.

As a result of this integration, we can display a rich combination of data, coming from different sources in different formats [23]. This application has been live for more than a year and can be accessed over the Web at: http://194.141.1.61/.

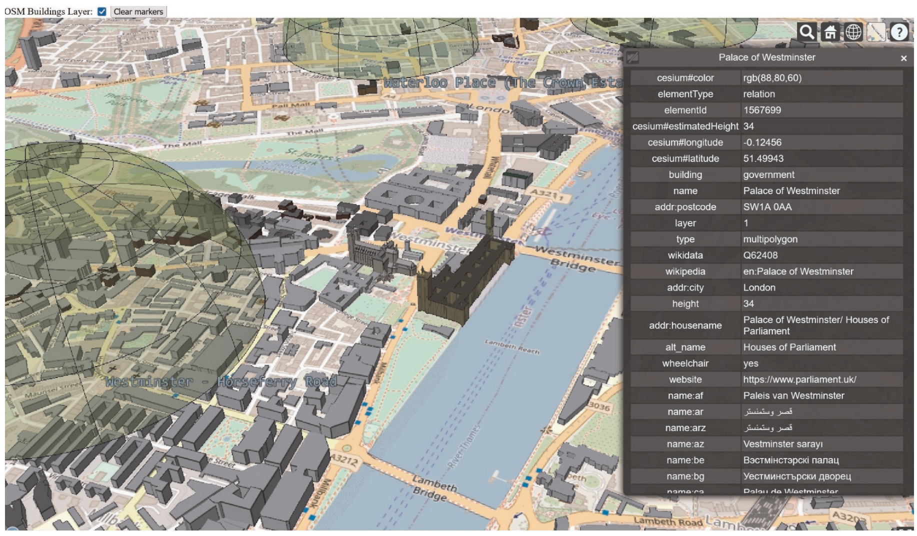
In this pilot project, we also added several components for platform management, implemented using public domain and community edition software:

- Remote access: Proxmox
- Identity management and access control: NGinx
- Indexing and searching: Elasticsearch
- Auditing and reporting of data services and operations: Kibana

The main data sources of these components are the system logs generated on different levels of operation of the platform software – OS, VMs, containers, servers, and runtime engines (see **Table 4**). These additional components are application independent; their primary role is to enhance the control of the operations. This way, they prepare the migration of the platform pilots to commercial provision, as well as the use of the platform as data and service provider in future dataspaces [4].

4.3 Impact of environment pollution on public health (London pilot)

This project started as a mirror of the project for monitoring the air pollution in Sofia. We first implemented most of the functionality we had in Sofia, this time on the private cloud of the Cyber Security Research Centre of London Metropolitan University (see **Figure 10**). The real-time sensor data we were ingesting from the stations in London came from 130 locations, so the amount of data was around 10 times bigger than in Sofia, although, in terms of real-time, it is still manageable. Because of this, the sensor data was initially gathered in the MongoDB database in JSON format for preliminary analysis. Further, we were able to collect offline archived data from the last 5 years in a structured format (CSV files), which was transferred to the server and stored initially in PostgreSQL database for trends analysis. Finally, we implemented our own sensor station to collect information about indoor pollution and used its output to correlate the indoor and outdoor factors in the area around the building of the university [26]. The correlation



Palace of Westminster

cesium#color	rgb(88,80,60)
elementType	relation
elementId	1567699
cesium#estimatedHeight	34
cesium#longitude	-0.12456
cesium#latitude	51.49943
building	government
name	Palace of Westminster
addr:postcode	SW1A 0AA
layer	1
type	multipolygon
wikidata	Q62408
wikipedia	en:Palace of Westminster
addr:city	London
height	34
addr:housename	Palace of Westminster/ Houses of Parliament
alt_name	Houses of Parliament
wheelchair	yes
website	https://www.parliament.uk/
name:af	Paleis van Westminster
name:ar	قصر وستمنستر
name:arz	قصر وستمنستر
name:az	Vestminster sarayı
name:be	Вэстмінстэрскі палац
name:bg	Уестминстърски дворец
name:ca	Palau de Westminster

Figure 10.
Data integration and visualization of the air pollution in London.

showed a very close dependence between indoor pollution and outdoor pollution with some rare outburst which activities inside the building can explain.

Further expansion of the platform was achieved thanks to the addition of a new data source with data about medication prescriptions, freely available from one of the sites of UK National Health Service (NHS). Due to the volume of this data, we transferred the entire dataset we obtained from NHS using NiFi directly to the Hadoop cluster. To enable further use and more complex analysis of all data periodically all the temporary data stored in MongoDB was transferred to the Hadoop cluster using NiFi. The two datasets – the air pollution readings and the prescription data – were then joined in Hadoop and analyzed using Spark for possible correlation between the pollution and the respiratory diseases, which were categorized on the base of the prescribed medications. The analysis was performed using three different methods of calculating the correlation – Pearson, Spearman and Kendall, with comparable results. **Figure 11** presents some of the results of this analysis for a fragment of the dataset of prescriptions against the category of respiratory diseases. As it is clearly visible from the diagrams, there is a strong correlation between most pollutants and respiratory diseases, with NO_2 predictably being the most harmful. This pilot has been live for several months and is available on the Web at: http://217.38.61.107:5000/.

The most significant addition to the platform in this project was the full utilization of Big Data technologies for cross-domain data analysis. The Hadoop cluster for managing the Big Data was created on three virtual machines, emulating hosts for one name node and two data nodes accommodating the data in Hadoop HDF memory space. The Big Data analytics was performed in a distributed environment with a Spark server operating on the name node and two Spark clients operating on the data nodes. The project is currently still under way and will continue analyzing the data for revealing further dependencies as well as the dynamic of changes depending on the environment conditions and the period.

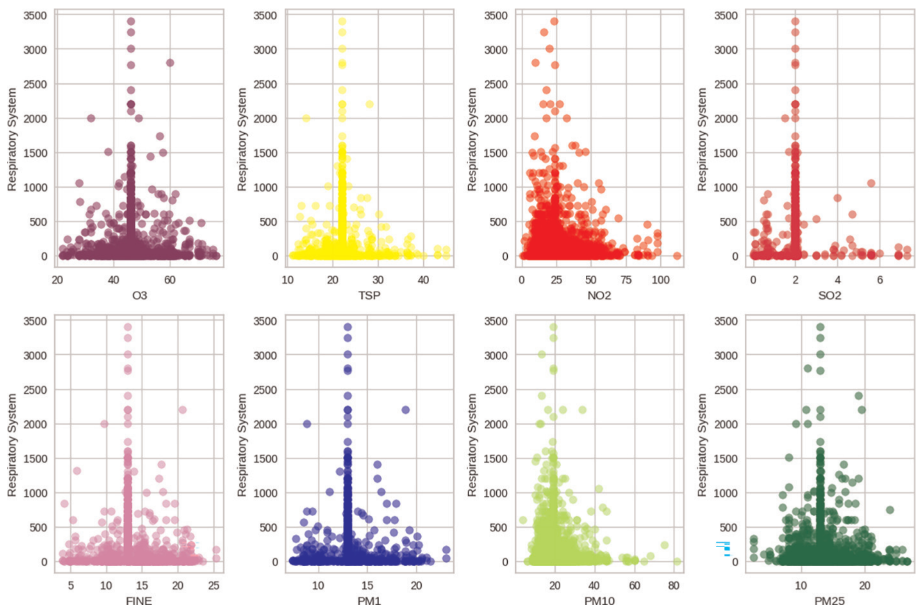


Figure 11.
Correlation between air pollutants and respiratory diseases in greater London.

Virtualization	Containerization	Orchestration
• *Heterogeneity* of hardware and system software	• *Modularization* of the software with no dependencies to set	• Support for *reusability* of existing solutions in process workflows
• *Scalability* of devices, memory and users	• *Efficiency* in memory, CPU and storage usage	• *Model-driven* application development
• *Choice* of convenient computational environment	• *Portability* of containers across platforms without code changes	• Support for *auditing* of monitoring, analyzing, and billing purposes
• *Transparency* of the physical location of the data and services	• Supporting *configuration generation* using templates	• *Reproducibility* of operations by preserving dependencies
• *Centralization* of the system administration and maintenance	• Full *traceability* of the operations for testing and debugging purposes	• Possibility for *process automation* based on planning heuristics

Table 7.
Advantages of cloud-based data platform for Big Data processing.

5. The gains and the burdens of being independent

Big Data platform on the cloud has many advantages for public and private businesses, large and small organizations, academia and industry (see **Table 7**).

Operationally, platforms on the premises provide more limited opportunities in comparison with the platforms of the public clouds. For example, most public clouds support the training of machine learning algorithms, which drastically reduces the *time* and *computational resources* needed for training. However, the private clouds retain the *data ownership* and protect the *privacy* of data and operations.

The software without license fees brings some disadvantages, too. Custom-built platforms may incur higher *maintenance costs* due to the need to hire highly qualified staff or even external consultants. But this drawback diminishes quickly with multiple projects because of the *cost spreading*, while the running costs on the public clouds are cumulative and, as a result, are higher per project. From a strategic perspective, in-house operation has an additional advantage – it stimulates the local economy by fostering *independence* from the commercial software vendors and service providers. In balance, this solution is definitively a better choice for project oriented organizations and software houses with extensive in-house development.

6. Possible extensions and directions for future work

The concept of a data platform for Big Data processing on a private cloud using free software proved itself strongly through the pilots. Our work in this direction can continue to make the platform more valuable for both the organizations which employ it as a development environment and for clients which rely on its services:

- Methodology adoption: The DevOps employs shared repositories for various purposes. Previously, we have used GitLab for automatic creation of computing infrastructure, component deployment and integration on the public cloud [20]. It significantly increases the productivity and improves the quality. It can be employed on the private cloud as well. This is valuable for software vendors and consultancy organizations in which multiple teams are working in parallel, sharing both data and software.

- Analysis hybridization: The open architecture allows the addition of more components, particularly for hybridization of the analytics. We have already done some work on hybrid analytics by combining pure data analysis with knowledge-based reasoning [16]. This would allow cross-domain analysis beyond the simple data analysis since it can utilize the dependencies within the data formulated in heuristic rules.

- Workflow automation: We have also conducted some experiments for the generation of Docker and Airflow, configuration files, based on the ontological model of the platform (see **Figure 3**). This can be done using templates, which can also be used for explanation generation, accounting for the causal links between tasks and results [21].

- Service homogenization: Combining bespoke and universal components may lead to granularity problems. For example, we were trying to implement the access control using Fiware Keyrock [27], but it required too many additional components, so we used Nginx instead. On the other hand, for the enrichment of data in tabular format, we wanted to use the **Grafterizer** [28], but for non-relational data, we needed other components, so we opted out in favour of a more universal bespoke development.

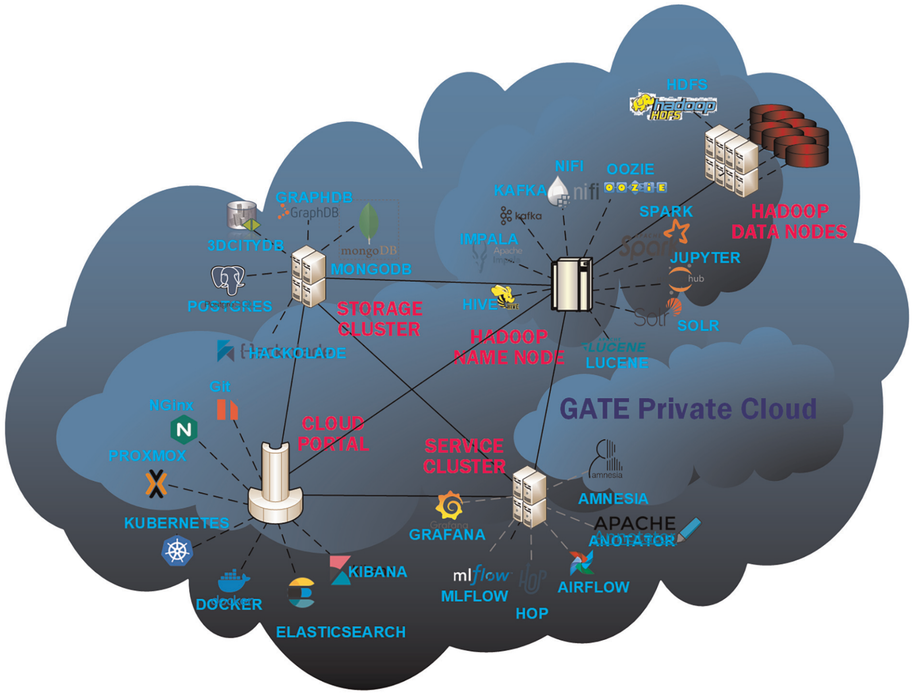


Figure 12.
Architecture of GATE big data platform.

With the growth of digital data, the need for utilization of data platforms becomes more important by the day. The open and highly scalable architecture of a platform built on private clouds using software without a license fee can easily evolve into an enterprise solution, running on multiple physical hosts and managing petabytes of data. This guarantees scalability without the need for significant initial investment. For example, **Figure 12** shows the enterprise version of the GATE Data Platform, built this way—not only a viable but also a highly lucrative option.

Acknowledgements

The work reported here has been conducted over a period of more than 5 years in collaboration with colleagues, students and industrial partners from London Metropolitan University in the United Kingdom and Sofia University in Bulgaria. It has been made possible thanks to several grants from the United Kingdom (Innovate UK and HEIF), the Bulgarian government (OPNOIR Grant Agreement No. BG05M2OP001-1.003-0002-C01) and the EU H2020 Research Framework (WIDESPREAD 2018–2020 Grant Agreement No. 857155). It has also been supported by private companies in the United Kingdom (Lloyds Banking Group, Oxagon) and Bulgaria (Rila Solutions, Ontotext), providing additional resources for platform development. The authors are greatly thankful to all collaborators and partners for their professional help, continuing support and unwavering trust over the years. All recommendations and considerations, however, are purely of the authors and should not be considered the official policy of these organizations.

Author details

Vassil Vassilev[1,2]*, Viktor Sowinski-Mydlarz[1,2], Pawel Gasiorowski[1,2], Sorin Radu[2], Sabin Nakarmi[1,2], Martin Hristev[1], Reza Baghaeishiva[1] and Tarun Bali[2]

1 Cyber Security Research Centre – London Metropolitan University, UK

2 GATE Institute – Sofia University, Bulgaria

*Address all correspondence to: v.vassilev@londonmet.ac.uk

IntechOpen

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

References

[1] Gartner, Inc. 10 top strategic technology trends [Internet]. 2023. Available from: https://www.gartner.com/en/information-technology/ [Accessed: July 06, 2023]

[2] Moses B, Gavish L. What is a data platform? [Internet]. 2023. Available from: https://www.montecarlodata.com/ [Accessed: July 07, 2023]

[3] Strong A. Containerization vs. virtualization: What is the difference? [Internet]. 2022. Available from: https://www.burwood.com/blog-archive/ [Accessed: July 07, 2023]

[4] Anjomshoaa A et al. Data platforms for data spaces. In: Curry E et al., editors. Data Spaces. Cham: Springer; 2022. DOI: 10.1007/978-3-030-98636-0_3

[5] IBM. IBM storage scale Big Data and analytics support [Internet]. 2023. Available from: https://www.ibm.com/docs/en/storage-scale-bda [Accessed: July 07, 2023]

[6] Hewlett-Packard Enterprise. HPE Ezmeral Data Fabric [Internet]. 2023. Available from: https://www.hpe.com/us/en/hpe-ezmeral-data-fabric.html [Accessed: July 07, 2023]

[7] Oracle. Oracle Big Data Appliance [Internet]. 2023. Available from: https://docs.oracle.com/en/bigdata/big-data-appliance/index.html [Accessed: July 07, 2023]

[8] Amazon Web Services, Inc. Amazon EMR [Internet]. 2023. Available from: https://aws.amazon.com/emr/ [Accessed: July 07, 2023]

[9] SAP. SAP HANA Cloud [Internet]. 2023. Available from: https://www.sap.com/uk/ products/technology-platform/hana.html [Accessed: July 07, 2023]

[10] Cloudera, Inc. Cloudera Data Platform [Internet]. 2023. Available from: https://www.cloudera.com/products/cloudera-data-platform.html [Accessed: July 07, 2023]

[11] Kunigk J, Buss I, Wilkinson P, George L. Architecting Modern Data Platforms. 1st ed. Sebastopol: O'Reilly; 2019. p. 640

[12] Amazon Web Services, Inc. AWS Lake Formation [Internet. 2022. Available from: https://aws.amazon.com/lake-formation/?c=a&sec=uc3 [Accessed: July 07, 2023]

[13] Google. Cloud data warehouse to power your data-driven innovation [Internet]. 2023. Available from: https://cloud.google.com/bigquery/ [Accessed: July 07, 2023]

[14] Microsoft. Azure Databricks [Internet]. 2023. Available from: https://azure.microsoft.com/en-gb/products/databricks [Accessed: July 07, 2023]

[15] Almeida A, Brás S, Sargento S, Pinto FC. Time series big data: A survey on data stream frameworks, analysis and algorithms. Journal of Big Data. 2023; **10**(1):83. DOI: 10.1186/s40537-023-00760-1

[16] White T. Hadoop. 4th ed. Sebastopol: O'Reilly; 2015. p. 754

[17] Taylor D. Top 15 Big Data tools and software [Internet]. 2023. Available from: https://www.guru99.com/big-data-tools.html [Accessed: November 07, 2023]

[18] Chambers B, Zaharia M. The Definitive Guide. 1st ed. Sebastopol: O'Reilly; 2018. p. 603

[19] Vassilev V, Sowinski-Mydlarz V, et al. Intelligence graphs for threat intelligence and security policy validation. In: Bansal P et al., editors. Intelligent Systems and Computing. Vol. 1164. Springer; 2020. pp. 125-139. DOI: 10.1007/978-981-15-4992-2_13

[20] Vassilev V, Phipps A, Lane M, et al. Two-factor authentication for voice assistance in digital banking using public cloud services. In: Proc. 10th Int. Conf. Confluence. Noida, India: IEEE; 2020. pp. 404-409. DOI: 10.1109/Confluence47617.2020.9058332

[21] Vassilev V, Ilieva S, Sowinski-Mydlarz V, et al. AI-based hybrid data platforms. In: Curry E et al., editors. Data Spaces. Springer; 2022. pp. 147-170

[22] Vassilev V, Ouazzane K, Sowinski-Mydlarz V, et al. Network security analytics on the cloud: Public vs. private case. In: Proc. 13th Int. Conf. Confluence. Noida, India: IEEE; 2023. pp. 151-156. DOI: 10.1109/Confluence56041.2023.10048889

[23] Vassilev V, Sowinski-Mydlarz V, Mariyanayagam D, et al. Towards first urban data space in Bulgaria. In: Proc. IEEE Int. Smart Cities Conference. Paphos, Cyprus: IEEE; 2022. pp. 1-7. DOI: 10.1109/ISC255366.2022.9922237

[24] Vassilev V, Virdee B, Ouazzane K, et al. Data platform and urban data services on private cloud. In: Zghang Y et al., editors. Smart Trends in Computing and Communications. Vol. 650. Springer LNNS; 2023. pp. 263-275. DOI: 10.1007/978-981-99-0838-7_23

[25] Cesium GS, Inc. The platform for 3D geospatial [Internet]. 2023. Available from: https://cesium.com/ [Accessed: November 07, 2023]

[26] OpenStreetMap Foundation. Planet OSM [Internet]. 2023. Available from: https://planet.openstreetmap.org/data [Accessed: November 07, 2023]

[27] Fiware. Keyrock Identity Manager [Internet]. 2023. Available from: https://keyrockfiware.github.io/ [Accessed: July 14, 2023]

[28] Stiftelsen S. Grafterizer 2.0 [Internet]. 2023. Available from: https://www.eubusinessgraph.eu/grafterizer-2-0/ [Accessed: November 07, 2023]