ArchaDIA: An Architecture for Big Data as a Service in Private Cloud

Marco Antônio de Sousa Reis and Aletéia Patrícia Favacho de Araújo
Department of Computer Science, University of Brasília, UnB, Brasília, Brazil

Keywords: Big Data, Cloud Computing, NoSQL, Hadoop, Data Engineering.

Abstract: There are multiple definitions and technologies making the path to a big data solution a challenging task. The use of cloud computing together with a proven big data software architecture helps reducing project costs, development time and abstracts the complexity of the underlying implementation technologies. The combination of cloud computing and big data platforms results in a new service model, called Big Data as a Service (BDaaS), that automates the process of provisioning the infrastructure. This paper presents an architecture for big data systems in private clouds, using a real system to evaluate the functionalities. The architecture supports batch/real-time processing, messaging systems and data services based on web APIs. The architectural description defines the technology roadmap, composed exclusively of big data tools. The results showed that the proposed architecture supports the facilities of cloud computing and performs well in the analysis of large datasets.

1 INTRODUCTION

The infrastructure required to support the demand for technology in modern life is complex, has a high financial cost and needs a specialized workforce, since the datacenter of the companies is usually heterogeneous, with multiple operating systems, storage devices, programming languages and application servers. In this context, cloud computing aims to streamline provisioning and optimize the use of datacenter equipment through virtualization, enabling better utilization and decreasing the idleness of computing resources (Mell et al., 2011).

The need to process and store a huge amount of data has became known as ”big data”, a concept that is related to the generation or consumption of a large volume of data in a short time, so that the traditional technology infrastructure can not process efficiently and at low cost (Chang, 2015a).

The value of combining these two trends, big data and cloud computing, has been recognized an is of interesting in the software industry and academia, leading to the creation of a technology category called Big Data as a Service (BDaaS) (Bhagattjee, 2014). But creating big data systems is not trivial, and organizations that need to use their private cloud have difficulty delivering effective solutions because of the company's lack of expertise.

To meet this demand, this paper presents an architecture for building big data systems in private cloud, called ArchaDIA (Architecture for Data Integration and Analysis), detailing the functionalities, techniques and tools most suitable for this type of system. The architecture uses the capabilities of cloud computing to shorten the time needed to build big data systems that operate in the same scenarios described in this proposal. As a result, architecture can help to reduce the time required to deploy big data solutions, avoiding the time spent in the early stages of adopting new technologies.

The main contributions of this paper are: (i) the formal and up-to-date architectural description for big data systems (Rozanski and Woods, 2012), (ii) a clear definition of the Big Data as a Service model, (iii) the description of techniques for creating data-intensive systems, (iv) architecture evaluation through a proof of concept (PoC) involving a real situation and real data.

The rest of the paper is organized as follows: Section 2 presents the concepts of big data and cloud computing. Section 3 gives the proposed architecture. Section 4 describes techniques for building big data systems. Section 5 discusses the methodology used to conduct the study, to evaluate the architecture and to build the proof of concept. In the end, Section 6 highlights the conclusion and some future works.

Reis, M. and Favacho de Araújo, A.
ArchaDIA: An Architecture for Big Data as a Service in Private Cloud.
DOI: 10.5220/0007787801870197
In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)*, pages 187-197
ISBN: 978-989-758-365-0

2 BACKGROUND

2.1 Cloud Computing

Cloud computing, according to (Foster et al., 2008), is a large-scale, scale-driven, distributed computing paradigm in which a set of resources is delivered on demand to external users on the Internet. This set of resources consists of computational power, storage, platforms and services.

Cloud computing differs from other models by being massively scalable, virtualized, encapsulated at different levels of service to the external customer, and by their services being dynamically configured. The essential features of cloud computing embodied in ArchaDIA, the architecture proposed in this article, are: (i) on-demand self-service; (ii) broad network access; (iii) resource pooling; (iv) elasticity; and (v) measured service.

2.2 Big Data

The term ”big data” has several definitions, depending on the context it applies. The first definition presented is (Chang, 2015a) and says that big data consists of large datasets with the characteristics of volume, variety, speed, and/or variability that require a scalable architecture for efficient storage, manipulation, and analysis. Also in (Chang, 2015a), big data refers to the inability of traditional data architectures to efficiently manipulate new datasets, forcing the creation of new architectures, consisting of data systems distributed in independent, horizontally-coupled computing resources to achieve scalability, using massively parallel processing.

A big data system is made up of functionalities to handle the different phases of the data life cycle from birth to disposal. From the point of view of systems engineering, a big data system can be decomposed into four consecutive phases, namely generation, acquisition, storage and data analysis, as listed in (Hu et al., 2014). **Data generation** refers to the way data is generated, considering that there are distributed and complex sources, such as sensors and videos. **Data acquisition** is the process of obtaining the data. This process includes partitioning into specific collections, data transmission and preprocessing. **Data storage** refers to data retention and management capabilities. The last phase of **data analysis** presents new methods and tools for querying and extracting information from datasets.

This paper adds the **data integration** phase, so that external systems can consume the data through web services, an approach that guarantees low coupling between the big data system and the data consumers.

2.3 Big Data as a Service (BDaaS)

Big Data as a Service (BDaaS) is a new model that combines the capabilities of cloud computing with the processing power of big data systems to deliver data, database, data analysis, and processing platform services, in addition to the tradicional service models (Paas, Saas and IaaS).

BDaaS represents an abstraction layer above data services, so the user selects the functionality, and the underlying infrastructure is in charge of provisioning, installing, and configuring the services, which are complex tasks that require specialist knowledge. In this model it is possible to rapidly deploy big data systems, reducing development time and cost in the early stages of the project's lifecycle (Zheng et al., 2013).

The implementation of big data systems involves high costs for configuring the infrastructure and obtaining skilled labor. A BDaaS framework can therefore help organizations move quickly from the starting point, which is big data technology research, to the final phase of the solution deployment. Even the phases involving pilot projects would be streamlined with the use of cloud-based technologies (Bhagattjee, 2014).

BDaaS includes other service models to address the specific demands of big data systems. A BDaaS cloud infrastructure must offer the following functionalities:

- **Data as a Service** (DaaS): refers to the availability of data sets through web services and it is implemented by the Application Programming Interface (API). The data services are independent of each other and are reusable;
- **Database as a Service** (DBaaS): refers to the provisioning of NoSQL databases. Although technically possible, the ArchaDIA, proposed in this work, does not provision relational databases management system (RDBMS) instances;
- **Big Data Platform as a Service** (BDPaaS): refers to the provisioning of big data clusters to run Hadoop or Spark[1]. The private cloud platform is responsible for installing and configuring the software, reducing the complexity of the management of big data environments;

[1] https://spark.apache programs.org/

- **Analytics as a Service** (AaaS): refers to the provisioning of data analysis tools from big data clusters. For AaaS, the tools are Hive[2] and Spark. Hive is a data warehouse tool with SQL support. Spark has a framework for in-memory processing that combines capabilities for processing SQL queries, real-time, machine learning and graphs;
- **Storage as a Service** (StaaS): refers to storage provisioning through distributed file systems (DFS). The service is implemented in two ways: (i) with OpenStack Swift[3] in the form of object storage (Varghese and Buyya, 2018) or (ii) with HDFS[4] from a Hadoop cluster. The techniques to select between one technology and another are described in Section 4.

Well-known cloud providers such as Amazon, Microsoft and Oracle already offer the BDaaS cloud model. In these cases, the provider is responsible for the equipment, the software instalation and configuration, the datacenter operation and the big data services.

Thus, as will be presented in Section 3, the ArchaDIA presents an alternative in which the BDaaS model uses its own infrastructure without the public cloud. This article presents the advantages of this approach that uses the big data in the private cloud.

2.4 Private Cloud

According to (Mell et al., 2011), the private cloud is one of the forms of deployment in cloud computing, in which computing resources are available only to an organization and its consumers. It may belong to, be managed, located and operated by the organization itself or by an outsourced company or even some kind of combination between them.

The private cloud is adequate when the company has the datacenter itself and its data demands a high level of security, as in the government, banking and telecom. It is not necessary for the datacenter to be large, so only a few servers can justify its adoption. In the private cloud computing resources are not necessarily available to the public.

The tool used to deploy the private cloud in this study is OpenStack[5], which is a widely used and tested open source IaaS tool. OpenStack supports the most important virtualization solutions such as KVM[6], Hyper-V[7], QEMU[8] etc.

2.5 Related Work

The correct selection of architecture components has the potential to reduce project costs, development time and abstracts the complexity of the underlying implementation technologies. In this direction, there are several architectures that can be used for big data solutions.

The Lambda Architecture, proposed by (Marz and Warren, 2015), was designed based on the principles of scalability, simplicity, immutability of data, batch and real-time processing frameworks. The architecture was created by observing the problems presented by traditional information systems, such as the complexity of the operation, the addition of new functionalities, the recovery of human errors and the optimization of performance.

This architecture uses big data techniques and tools to process and store data, including technologies for batch processing, the NoSQL database for data management, and the messaging system for data ingestion. The goal is to combine the benefits of each technology to minimize the weaknesses. In order to organize the internal elements, the architecture is divided into three layers, which are:

- Batch layer: stores the main copy of the data and preprocesses the batch views with the batch processing system (Hadoop);
- Serving layer: stores the result of batch processing in a data management system for queries, such as a NoSQL database;
- Speed layer: processes incoming data while batch processing, ensuring the execution of the query with real time data.

In (Chang, 2015b) the authors show a reference architecture for big data solutions. The goal is to create a conceptual model of architecture for big data architecture, without reference to specific technologies or tools. In this model the following functional logical components are defined:

- System Orchestrator: defines and integrates activities into a vertical operating system. It is responsible for setting up and managing the other components, or directly assigning the workload to the computational resource;

[2]https://hive.apache.org/
[3] https://docs.openstack.org/swift/latest/
[4] http://hadoop.apache.org/
[5]https://www.openstack.org
[6]https://www.linux-kvm.org
[7]https://www.microsoft.com/en-us/cloud-platform/server-virtualization
[8]http://www.qemu.org/

- Data provider: includes new data or sources of information in the system;
- Big data application provider: encapsulates business logic and functionality to be executed by the architecture. It includes activities such as collection, preparation, analysis, visualization and access to data;
- Big Data Framework Provider: consists of one or more technologies to ensure flexibility and meet the requirements that are set by the big data application provider. It is the component that gets the most attention from the industry;
- Data consumer: the end users and other systems that use the result produced by the big data application provider.

In (Bhagattjee, 2014), the author defines that BDaaS is a distributed, horizontally scalable, cloud-based computing framework designed to handle large datasets (big data). However, due to the number of technologies available, it is difficult to identify the right solutions for each demand. As a result, the development of big data systems ultimately involves high costs both for infrastructure management and for skilled labor. As a proposed solution, (Bhagattjee, 2014) introduces a framework to help technology users and suppliers identify and classify cloud-based big-time technologies. The framework is described in layers, each with a set of responsibilities and the appropriate tools for implementation.

3 ARCHITECTURE FOR DATA ANALYSIS - ArchaDIA

This section introduces the design of the big data architecture to ensure the state of the art in integrating big data resources and cloud computing. The proposal uses and extends the models defined by (Bhagattjee, 2014) (Chang, 2015b) (Marz and Warren, 2015).

The architecture's **scope** includes support for two processing modes: batch and real-time. The batch processing platform will be used for analysis in the complete dataset, in which the need to access its result has no rigid time limitation, i.e. it is possible to wait minutes or hours for the result. The real-time processing platform will be used in applications with constant data flow, that is, data is entered continuously and the system response must be immediate.

The design of big data systems is more complex than traditional projects, mainly because it involves distributed processing. The difficulties include not only the technologies but also the processing and storage techniques that must be reviewed in this new context of data-intensive applications. In order to guide the creation of this type of system, the study of (Chen and Zhang, 2014) proposes seven **principles**:

1. Good architecture and good frameworks - there are many distributed architectures to big data and each uses different strategies for real-time and batch processing;
2. Support for various analysis methods - data science involves a large number of techniques that need to be supported by new architectures, such as data mining, statistics, machine learning etc;
3. No one size fits all - there is no single solution that suits all situations, since each technology has limitations. One should choose the right tool for each technique and situation;
4. The analysis must be close to the data - the processing must have high performance access to the storage, which favors the use of data lakes, as detailed in Item 4.3;
5. Processing must be distributable for in-memory analysis - Massively Parallel-Processing (MPP) is one of the bases of big data systems, in which data is accumulated in the datacenter storage system, but must be partitioned to allow parallel processing;
6. Storage must be distributable to memory retention - MPP tools often divide data into blocks in memory;
7. A mechanism is required to coordinate data and processing units to ensure both scalability and fault tolerance.

As a **constraint**, the tools used in ArchaDIA should be free and open source (FOSS) to ensure that the solution can be used in government agencies or private companies without the limitations of the cost of acquisition.

The **stakeholders** are people and organizations interested in the architecture. Thus, for the architecture of big data systems, those interested are: (i) Data Scientists: who perform *ad hoc* queries and data analysis; (ii) Software Developers: who create the systems; (iii) Enterprise Systems: production systems and databases in the organization; (iv) External Systems: systems and databases in operation outside the organization; (v) Infrastructure Administrators: responsible for maintaining the datacenter environment, including servers, storage, network and database.

3.1 Architectural Views for Big Data Systems

Architectural views are used to show different aspects of the big data system in an abstract way without technical details. The formalization of the documentation uses the (1) context, (2) functional and (3) deployment views, a combination that clearly illustrates big data in the ArchaDIA.

3.1.1 Context View

It presents the architecture from a conceptual perspective, in which the operational environment of the project is illustrated and shows what is inside and outside the boundaries of the BDaaS architecture, as verified in Figure 1. On one side are the data sources, in the center the private cloud and the BDaaS, and on the right side are the users of the system.

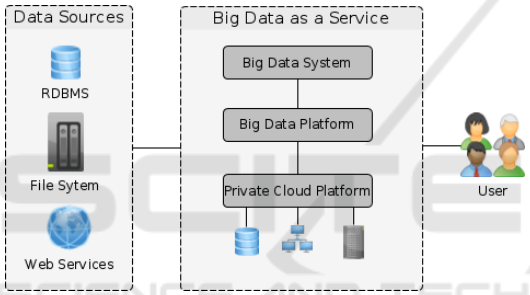

Figure 1: ArchaDIA Context View.

The data sources are composed of pre-existing relational databases, file systems, or web services in the organization. After importing these records into the big data system, data is available for processing or storage and then accessible to users.

3.1.2 Functional View

Describes the uses, components, interfaces, external entities and the main interactions between them. Following this definition, Figure 2 shows the Functional View of the architecture through a component diagram.

The **Data Source** component is an external entity that represents, as the name suggests, any mechanism that provides corporate data, and includes the RDBMS, file system, and web services.

The **Big Data ETL** performs the processing required to convert the data from its source format to the formats supported by the big data storage engine. This component and the techniques used are detailed in Section 4.

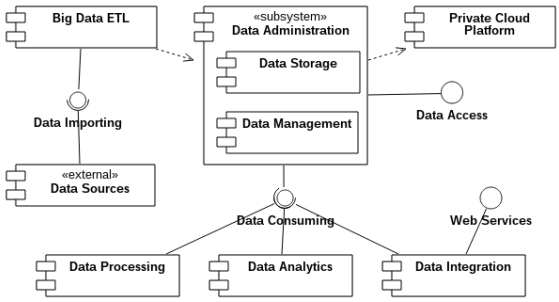

Figure 2: ArchaDIA Component Diagram.

The **Data Administration** subsystem is a component for managing the data lifecycle in the organization. It consists of two components. The first one is the **Data Storage** service, responsible for persisting the data in the distributed file system. The component of **Data Management** consists of the NoSQL database.

The **Data Processing** component consists of the mechanisms for batch and real-time processing, as well as support for the implementation of big data programs such as Hadoop and Spark, by the use of their respective frameworks.

The **Data Integration** is a feature present in the latest big data architectures and provides an API for external systems to insert and query system data with high performance. The API is built by a NoSQL database and available through web services, a trend in the area of Big Data and Cloud Computing that is detailed in Section 4.

The last component is the **Data Analysis**, which combines *ad hoc* query mechanisms, statistics, and machine learning algorithms. These features are used by data scientists and are part of an area known as big data analytics.

3.1.3 Deployment View

Describes the environment where the system will be installed, the hardware and the software necessary for its execution, that is, in this view the necessary equipment types, software and basic network requirements are defined to implement the big data system. This view shows two diagrams, one for deploying the private cloud platform and one for deploying the big data systems. Figure 3 shows the necessary equipment for private cloud deployment.

In ArchaDIA, big data systems are deployed through the private cloud platform, as can be seen in Figure 4. The servers represented in the diagram can be VMs or physical servers (bare metal). For the creation of PoCs it is possible to use a single server for

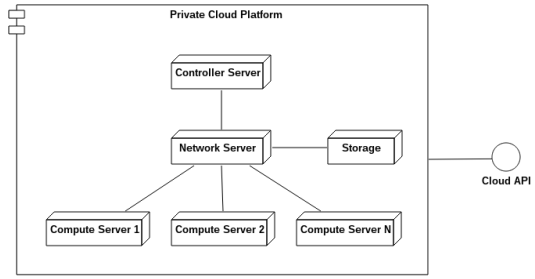

Figure 3: Private Cloud Deployment Diagram.

the private cloud platform and for the big data systems, however, the performance can not be evaluated because the computational resources are limited.

The functionalities provided in the architecture to meet the demands of the big data systems are (i) **Big Data Cluster**, where the data storage and analysis services reside; the (ii) **API Server**, which encompasses the **Data Integration** service; (iii) **NoSQL Server** includes **Data Management**; and finally, (iv) the **Storage** offers the **Data Storage** service with object storage, that is, without the need of a big data cluster. The Table 1 lists these components and their implementation tools.

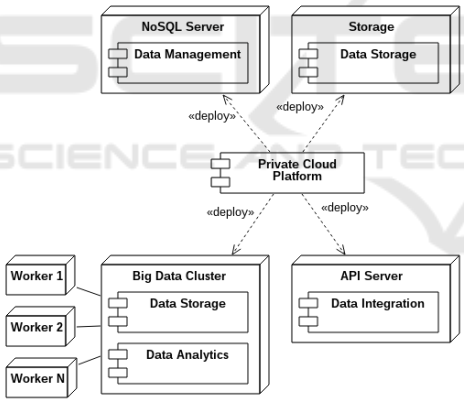

Figure 4: Deployment Diagram for Big Data Systems.

3.2 Layered Implementation

The ArchaDIA architecture obeys the layered architectural style, in which the structure is divided into logical modules, each with a well-defined functionality. This view serves the non-technical stakeholders, as it presents the description of the functionalities without details of implementation or technologies. Figure 5 shows the layers that are detailed throughout this section. The layers are functionally independent of each other, are low-coupling and the communication between them is done with web services and APIs.

Table 1: Software Components Dependencies.

Component	Requirement
Private Cloud Platform	OpenStack Pike CentOS 7 KVM
Big Data ETL	Java 8 Apache Sqoop[9] Apache Flume[10] Apache Kafka[11]
Data Storage	Apache HDFS OpenStack Swift
Data Management	Apache Cassandra Apache HBase
Data Processing	Apache Hadoop Apache Spark Cloudera Hortonworks
Data Integration	Spring Boot 2.0
Data Analysis	Apache Hive Apache Spark Hue

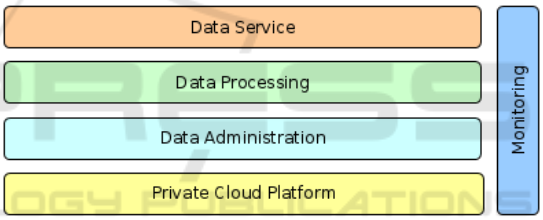

Figure 5: ArchaDIA Layered View.

The **Data Service Layer** provides a data access interface through a flexible, loose coupling communication mechanism with external systems. This layer is related to the DaaS model. This layer is shown in the Figure 6. The data flow starts with the External System request to the Web API and has two ways: (i) to ingest the data in the Messaging System, in an assyncronous way; and (ii) NoSQL query. It should be noted that operations with the Messaging System are unidirectional, since the purpose of this proposal is to allow the insertion of new records as an alternative to improve performance (Chang, 2015b).

The **Data Processing Layer** offers a platform that allows the user to execute big data programs and data analysis, including SQL queries. The layer is formed by BDPaaS and AaaS.

The **Data Administration Layer** offers storage and data management services. Thus, this layer is composed of DBaaS and StaaS. Data is stored permanently or temporarily, according to user demand. Security, access control, integrity, replication, and scal-

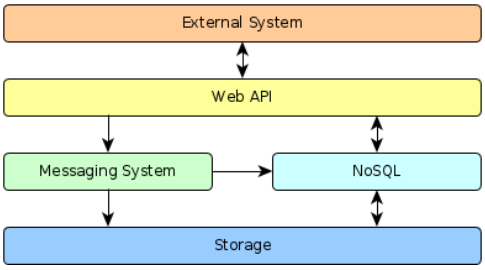

Figure 6: Data Integration Diagram.

ability are provided by the cloud platform.

The **Private Cloud Platform Layer** is responsible for the management of computing resources (processing, memory, storage and networking). The user can start or stop the services through the web interface or console, without direct access to the underlying hardware resources or deployment technologies.

The **Monitoring Layer** checks the operating conditions and usage of the systems in the datacenter. Users of this layer have access to metrics on resource availability and utilization. Collecting these metrics allows datacenter administrators to plan scalability of systems. The objective of the layer is related to the quality of the service offered, as it allows the monitoring of failures, unavailability, underutilization and resource overload.

4 TECHNIQUES FOR BUILDING BIG DATA SYSTEMS

The description of an architecture should detail the best practices for building systems, which is especially important in an area as recent and complex as big data. The construction of big data systems based on computational cloud has specific techniques for sizing, loading, storage, modeling and data integration.

The deployment of infrastructure for big data systems demands great effort from technology teams. The difficulties include: (i) the installation and configuration of the big data cluster and NoSQL databases; (ii) sizing the resources and attend to changes in the processing demand; and (iii) the provisioning of data services.

4.1 Resource Sizing

Resource sizing for big cloud data systems is the subject of several researches (Corradi et al., 2015), including the use of predictive algorithms and the automatic provisioning of Hadoop clusters. In many companies it is common to find large clusters with dozens of servers, however, this type of installation tends to be oversized to meet processing peaks.

With ArchaDIA, the recommendation to meet the demand for large-scale processing is to use several smaller big data clusters, one for each type of workload, since most Hadoop jobs run on datasets with less than 100 GB (Appuswamy et al., 2013). In these cases, a cluster with up to three nodes and 48 GB memory can be used as a starting point. After the processing is finished, the resources can be released. For a NoSQL database the starting point is a single VM with 16 GB memory. In this case, the resources are not released, since the duration of this processing is undetermined.

4.2 Cloud Storage

In the early versions of Hadoop, data analysis was performed using data from the cluster's file system, because HDFS is optimized for this purpose. However, for cloud-based systems, this approach is not the most efficient and durable. As the data is directly connected to the cluster, there are limitations in or even the impossibility of using the cloud characteristics. For example, considering a Hadoop cluster in the datacenter, if the user needs more disk space, the storage capacity can not be easily increased because only the datacenter operations team has this capability. Similarly, when the cluster is released, its data is usually deleted. To make further analysis on this deleted dataset, the data sources need to be copied back to a cluster.

One possible solution to this problem is the use of object storage technology, separating the data from the processing. The object storage (ObS) or object storage device (OSD) stores the data as objects of variable size, unlike traditional block storage (Factor et al., 2005). Thus, object storage features are: durability, high availability, replication, and ease of elasticity, allowing storage capacity to be virtually infinite. In object storage each stored item is an object, defined by a unique identifier, offering an alternative to the block-based file model.

Because of these facilities, storing data in the cloud can be done through object storage. Following this trend, leading cloud providers have their object storage implementations, such as AWS S3[12], Oracle Object Storage[13], Azure Blob Storage[14] and Google

[12]https://aws.amazon.com/s3/

[13]https://cloud.oracle.com/storage/object-storage/features

[14]https://azure.microsoft.com/en-us/services/storage/blobs/

Cloud Storage[15]. In Openstack, the object storage module is Swift (Rupprecht et al., 2017), in which data analysis can be performed with the ArchaDIA architecture.

4.3 Data Lake

Data lakes are centralized repositories of enterprise data, including structured, semi-structured and unstructured data. This data is usually in its native format and stored on low-cost, high-performance file systems such as HDFS or object storage (Dixon, 2010). The purpose of the data lake is different from a data warehouse (DW). In DW, the data are processed and structured for the query and the structure is defined before ingestion in the system, through ETL routines. This technique is called schema-on-write, a task that is not technically difficult, but is time-consuming.

In data lakes the data is in its original format, with little or no transformation and the data structure is defined during its reading, a technique known as schema-on-read. Users can quickly define and redefine data schemas during the process of reading the records. With this, the ETL runs from the data lake itself (Fang, 2015).

Data lake provisioning and configuration are performed by the private cloud platform, with the OpenStack Swift module. Swift is integrated with Hadoop and Spark in order to allow data analysis with the main file formats: SequenceFiles, Avro[16] and Parquet[17] (Liu et al., 2014).

The advantage of the data lake is its flexibility, which is at the same time a problem because it makes the analysis very complete, but also complex. Data lake users should be highly specialized, such as data scientists and developers. There are also other risks in adopting data lakes, such as quality assurance, security, privacy and data governance, which are still open questions.

4.4 NoSQL Databases

This new database paradigm, which does not follow relational algebra, is generally called Not Only SQL (NoSQL). In a NoSQL database, the data is stored in its raw form and the formatting of the result is done during the read operation, a feature called schema-on-read (Chang, 2015a).

NoSQL has fast access to read and write, supports large volumes of data and replication, so they are suitable for big data systems. However, NoSQL databases do not follow the same rules and standards as a relational database. For example, there is no native SQL support, and queries are typically run in proprietary languages, or through third-party tools.

At this point, there are big differences between relational and NoSQL modeling. While a relational data model is standardized to avoid data redundancy, NoSQL databases do not use normalization, and data is often duplicated in several tables to ensure maximum performance (Chebotko et al., 2015).

4.5 API Management

The use of web APIs is becoming the standard for web, mobile, cloud and big data applications (Tan et al., 2016). APIs make it easy to exchange data and are used to integrate business, make algorithms available, connect people, and share information between devices. This new business model, called the APIs economy, enables companies to become true data platforms, which simplifies the creation of new services, products and business models (Gartner, 2018).

Web APIs are composed of independent services in the form of reusable components, which can be combined to create the data platform. For example, a company can create a new service by using third-party APIs, such as maps, machine learning, geolocation, and payments. These services are usually based on REST and JSON, thus allowing the sharing of the data and the new features with high performance. This is the strategy adopted by major API providers and users such as Netflix, Google, AWS and eBay.

In this context, it is extremely important that a big data architecture provide technological support for API management. In ArchaDIA, the Data Integration Component is the technical solution for creating data services by accessing NoSQL databases or the Hadoop cluster. The API server is permanent and the VMs are not released, only resized in the case of processing peaks.

5 ARCHITECTURE EVALUATION

The evaluation of the proposed architecture (ArchaDIA) used a proof of concept (PoC), in which the usage scenarios and the behavior of the system were verified. In this way, it was possible to determine the positives and negatives of the project. After defining the functionalities of the BDaaS, experiments were conducted using techniques and tools to create big data systems in order to find the most appropriate combination.

[15] https://cloud.google.com/storage/docs/

[16] https://avro.apache.org/

[17] https://parquet.apache.org/

The Big Data Access Tool (BDAT)[18] is the practical implementation of PoC and was used to evaluate the capabilities of the architecture proposal in the form of a big data system. The BDAT was written in Java and incorporates frameworks for big data, ETL, API management and messaging system. Considering the diversity of technologies available in the big data area, BDAT represents an abstraction layer between the functionalities of a big data system and its implementation software, and can be used to create new big data systems.

The experiments used real data sources from the Brazilian government and the performance was measured in four situations: (i) batch/real-time processing of big data; (ii) *ad hoc* queries; (iii) ingestion of records in the system; and (iv) data query by API.

The dataset consists of several tables of systems available in the TJDFT, a Brazilian Court, totaling approximately 1.5 billion records that were imported from the enterprise RDBMS. The Hadoop/Spark cluster used in the experiments has four nodes, one master and three worker nodes, as shown in Figure 4. Evaluations were performed by simulating routine activities, such as executing SQL commands for extracting dataset information, such as those shown in the table 2. In the RDBMS the fields used for data consolidation are indexed and partitioned, at a high level of optimization. In the cluster analysis tools were used on files recorded in HDFS and object storage Swift.

The ArchaDIA involves the areas of Big Data, Cloud Computing and the intersection between them. Therefore, it was evaluated from different perspectives. Initially, it was evaluated as a reference architecture independent of technologies and implementations, in order to contribute to the research and development of big data systems. Finally, services, technologies, and how they relate to the private cloud environment are demonstrated.

5.1 Deployment Roadmap

The private cloud deployment used OpenStack and its specific modules that supports big data (Sahara) and databases (Trove). The installation scripts, commands, procedures, and configurations are available in the repository[19].

In addition to the Web interface, OpenStack offers the option of operating via command line, which was the option used in this study. After the complete environment configuration, you can provision a Hadoop cluster with a single command. With the cloud operating platform, the next step was provisioning services. The roadmap used for the creation of the PoC (Section 5.2) and for the initial data load using BDATconsists of the following steps:

1. Provision the big data cluster;
2. Provision an instance of the NoSQL Server (Cassandra);
3. Provision an instance of the Spring Boot API Server with BDAT;
4. List the available tables of the RDBMS environment;
5. Import each table to the big data staging area;
6. Convert imported files to Avro format and write them to HDFS and data lake;
7. Create the tables in Cassandra and load them with the files imported;
8. Perform the analyzes in the dataset with the Hadoop and Spark cluster;
9. Query through the Web API;
10. Release the cluster resources.

5.2 Proof of Concept

The first experiment was the analysis of the complete dataset with batch and real-time processing tools. The analyzes were performed with the execution of SQL commands in the Hadoop/Spark cluster and in the RDBMS. The second experiment was to write the records in the NoSQL database. Finally, the last simulated situation was the query of the records through the Web API. The operations available on the Web API are divided into three categories:

- Data Access: data inclusion, change and query operations;
- Data Store: lists the available tables in the PoC;
- ETL: data import and export operations, as well as list of available tables in the RDBMS.

In each experiment, we used three load levels: (i) low, with up to 10 million records; (ii) moderate, with up to 100 million records; and (iii) high, from 100 million records. Thus, the minimum amount of resources required to support the experiments was verified, avoiding oversizing or undersizing.

The proof of concept allowed us to verify that ArchaDIA supports the expected characteristics of the Big Data as a Service model.

The results in the Table 2 show that the correct combination of technologies and techniques for building big data systems in the cloud ensure performance

[18]https://github.com/masreis/big-data-access-tool

[19]https://github.com/masreis/big-data-as-a-service-openstack

similar to the traditional datacenter solutions. An important point to note is the performance and disk saving made possible by the new file formats compression, such as Avro[16].

6 CONCLUSION AND FUTURE WORKS

This study describes the functionalities and the proposed solutions for the big data area in the private cloud, adding new practical use cases evaluated with PoC in real scenarios. As a result, an architectural description was formalized, with its specific systems-building techniques in the form of a technology roadmap that can be used to deploy new solutions, or as a tool for communicating with non-technical users.

The study of the state of the art lead to conclude that the object storage is more interesting than HDFS in the cloud, since there is no great performance difference between the technologies. This point reinforces the importance of loose coupling in the proposed architecture, and is pointed as a trend along with the advancement of the data lakes.

Provisioning the big data cluster in the ArchaDIA takes a few minutes, as opposed to installing an RDBMS in a datacenter, which can take hours. The query by keys in NoSQL is not as fast as that of RDBMS, however it presents acceptable performance, considering that the NoSQL table was not as optimized as that of RDBMS in the experiments (Chebotko et al., 2015).

Table 2: Results of the Experiments.

Item	Small	Medium	Large
Cluster provisioning	160 sec.	180 sec.	190 sec.
Size of the dataset (Avro)	400 MB	4 GB	18 GB
Size of the dataset RDBMS	-	-	76 GB
Cluster data analysis	8 sec.	80 sec.	150 sec.
RDBMS data analysis	6 sec.	90 sec.	201 sec.
NoSQL query by key	0.03 sec.	0.06 sec.	0.1 sec.
RDBMS query by key	0.01 sec.	0.02 sec.	0.04 sec.

The Big Data and Cloud Computing research presents several open issues that will be considered in the future works (Varghese and Buyya, 2018) (Taleb and Serhani, 2017). The evolutions of ArchaDIA in the future include (i) the provision of a job completion prediction model; (ii) a pre-processing methodology to guarantee data quality and cleanliness in the analysis and integration phases; (iii) evolution of the disk load balancing mechanism, considering the imbalance between CPU and I/O; (iv) provide a security and data sharing model, considering a multi-user cloud; and (v) support for other resource managers (Kubernetes, Swarm and Mesos).

REFERENCES

Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., and Rowstron, A. (2013). Scale-up vs scale-out for hadoop: Time to rethink? In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 20. ACM.

Bhagattjee, B. (2014). *Emergence and taxonomy of Big Data as a service*. PhD thesis, Massachusetts Institute of Technology.

Chang, W. L. (2015a). Big Data Interoperability Framework: Volume 1, Definitions. *NIST special publication, Information Technology Laboratory, Gaithersburg*, 1.

Chang, W. L. (2015b). Big Data Interoperability Framework: Volume 6, Reference Architecture. *NIST special publication, Information Technology Laboratory, Gaithersburg*, 6.

Chebotko, A., Kashlev, A., and Lu, S. (2015). A big data modeling methodology for apache cassandra. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 238–245. IEEE.

Chen, C. P. and Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275:314–347.

Corradi, A., Foschini, L., Pipolo, V., and Pernafini, A. (2015). Elastic provisioning of virtual hadoop clusters in openstack-based clouds. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1914–1920. IEEE.

Dixon, J. (2010). Pentaho, Hadoop, and Data Lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/.

Factor, M., Meth, K., Naor, D., Rodeh, O., and Satran, J. (2005). Object storage: The future building block for storage systems. In *Local to Global Data Interoperability-Challenges and Technologies, 2005*, pages 119–123. IEEE.

Fang, H. (2015). Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 820–824. IEEE.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee.

Gartner (2018). Welcome to the API Economy. https://www.gartner.com/smarterwithgartner/welcome-to-the-api-economy/.

Hu, H., Wen, Y., Chua, T.-S., and Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *IEEE access*, 2:652–687.

Liu, X., Iftikhar, N., and Xie, X. (2014). Survey of real-time processing systems for big data. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, pages 356–361. ACM.

Marz, N. and Warren, J. (2015). *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co.

Mell, P., Grance, T., et al. (2011). The NIST definition of cloud computing.

Rozanski, N. and Woods, E. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley.

Rupprecht, L., Zhang, R., Owen, B., Pietzuch, P., and Hildebrand, D. (2017). Swiftanalytics: Optimizing object storage for big data analytics. In *Cloud Engineering (IC2E), 2017 IEEE International Conference on*, pages 245–251. IEEE.

Taleb, I. and Serhani, M. A. (2017). Big data pre-processing: Closing the data quality enforcement loop. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 498–501. IEEE.

Tan, W., Fan, Y., Ghoneim, A., Hossain, M. A., and Dustdar, S. (2016). From the service-oriented architecture to the web API economy. *IEEE Internet Computing*, 20(4):64–68.

Varghese, B. and Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861.

Zheng, Z., Zhu, J., and Lyu, M. R. (2013). Service-generated big data and big data-as-a-service: an overview. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 403–410. IEEE.