Taking an Electronic Ticketing System to the Cloud: Design and Discussion

Filipe Araujo, Marilia Curado, Pedro Furtado, Raul Barbosa
CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal
filipius@uc.pt, {marilia, pnf, rbarbosa}@dei.uc.pt

***Abstract*—In this paper we address the challenge of creating an electronic ticketing system for transportation systems that can partially or completely run on the cloud. This challenge is defined within the scope of an industrial project. The resulting system should be able to reach a large spectrum of customers and should provide two key advantages: lower operational costs, especially for small clients without IT departments, and faster execution of queries for monthly or other sorts of analysis, using the elasticity of cloud-based resources.**

To fulfill the goals of the project, we propose very standard technologies and procedures: a three-tiered architecture; a separation of the online and analysis databases; and an Enterprise Service Bus to get the input from very diverse hardware and software stacks. In this paper we discuss several options regarding the location of these facilities on the cloud and we also evaluate the costs involved.

While this work already defines many features of the system, it must be considered as preliminary, as some open details remain for future work.

***Index Terms*—Transportation systems, Electronic ticketing systems, Cloud computing**

I. Introduction

In this paper we address the challenge of taking the whole or part of the electronic ticketing service of the Octal company to the cloud. This effort involved the University of Coimbra and was financed by Portuguese and European public institutions under a project called "Electronic Ticketing as a Service". The movement to the cloud can offer two potential advantages: by reducing upfront investment and management costs, Octal should be able to reach a wider range of customers, namely small ones without their own IT resources; and exploring the elasticity of the cloud can speed up analysis operations that take a large computational effort.

Building a ticketing system is a really daunting task, due to the huge diversity of scenarios in real existing transportation systems and due to the myriad of subproblems to solve. For example, a major source of complexity comes from the need to share revenues. Since carrier companies may sell combined transportation titles allowing users to travel in multiple carriers in different segments of the same trip, the revenues of such trip must be divided. Another problem concerns the interaction with a wide range of access points for customers, vendors, managers, validation points, etc. I.e., ticketing systems must interact with a fairly large number of different hardware and software stacks. Our challenge is, therefore, to create a simple and elegant architecture that accommodates differences in software stacks and hardware using standard well-understood technologies.

For these reasons, we resort to a three-tiered architecture [2], with separate online and analysis databases. To accommodate customer's requests and needs, all parts of the system may run on the cloud or on-premises. For example, while running the application server(s) on the cloud may easily offer transparent elasticity, customers may prefer to keep their data on premises, and manually tune the application servers. To support all the legacy hardware of the carrier companies and enable reading and writing data to their own terminals we use an Enterprise Service Bus[1], as these offer a rich set of pull and push-based interfaces and transformation tools.

One of our significant efforts in this paper is to determine where and how to deploy the analysis database along with its functions. We evaluate a number of possibilities regarding data management, like, how to divide the data in the database, or how to deal with old data. We also discuss the best options to keep the data on the cloud (e.g., key-value storage versus a relational database), and how to move data from the online to the analysis database. This discussion is followed by an evaluation of the costs involved in cloud solutions, as this is a crucial point for the cloud adoption (or non-adoption). This comparison focuses on Infrastructure-as-a-Service (IaaS) versus Platform-as-a-Service (PaaS) costs, but also considers the possibility of using private resources.

We think that the architecture we are proposing is flexible enough to run either on-premises or on the cloud. Indeed, depending on the particular needs of the transport operators, we may outsource different components of the architecture. To ensure this property, and to avoid vendor lock-in, we propose using an IaaS. Our price comparisons demonstrate the feasibility of this option for a cloud-based ticketing system.

In the remainder of this paper we start by enumerating in Section II the most important ticketing solutions that use the cloud. In Section III, we propose a simple three-tier architecture for Octal. In Section IV we provide details regarding the management of the ticketing data. Then, in Section V, we approximately calculate the costs inherent to IaaS and PaaS solutions in some of the larger cloud players in the market. In Section VI, we conclude the paper and point out pending tasks that we must tackle for a successful closure of this project.

[1]See for example http://www.mulesoft.org or http://jbossesb.jboss.org.

II. Overview of Related Solutions

This section presents a selection of the cloud based ticketing solutions available on the Internet.

A. Atlas System for Cloud Ticketing

Xerox has developed a cloud based ticketing platform, the Atlas system[2], which is being used by over 300 transportation operators, with different dimensions and characteristics. For instance, this platform can cover a small dimension system with around 32 buses as in the urban area of Chalons-en-Champagne as well as the large transportation system in the Quebec metropolitan region, including 3000 buses, 5 railway lines and 4 metro lines. The Atlas system is also used in Riga and Peru.

The Atlas platform supports multimodal systems under multiple operators, in a flexible way and through a web interface. Data is organized in a database according to the different functionalities of the system, namely, accounting, marketing and fraud detection. Security is provided through access control and secure communication between the servers. The Simple Network Management Protocol (SNMP) [4] is extended with an additional level of authentication between hubs and devices. In addition, the data exchange with the systems located inside the transportation systems is protected with authentication through a double Virtual Private Network (VPN).

Equipment and workstations are controlled by a central server, which has a global view of the whole network. Each device has an individual access code that grants access to its physical and operational status. The Atlas platform supports the emission of alerts in real time, diagnostic and preventive measures, as well as remote management of applications. Operation continuity in emergency situations is guaranteed by a recovery plan that is triggered when such situations occur. However, the information provided on the Atlas website lacks details on the specific mechanisms used for recovery.

Rates can be defined according to multiple criteria, combining for instance transportation pass owners and special events. The rates are stored in the central system and sent to all the devices in a single operation, while the system is working.

The Atlas platform consists of a multiple layer architecture, using Oracle databases, Extract/Transform/Load (ETL) tools, and Oracle Warehouse Builder.

B. Accenture - Ticketing as a Service

Accenture presents a solution of Ticketing as a Service based on cloud services[3]. This platform supports multimodal systems under multiple operators. The interface is based on the web and it allows remote configuration and installation of devices as well as the remote configuration of rates in the devices. This product is being used in the Finish railway system.

The Accenture platform relies on a Microsoft system, but other popular systems in the mobile market such as Windows Mobile, iOS e Android are also supported.

C. Gemalto - transport transaction: e-ticketing, access control, data storage, and authentication

Gemalto targets ticketing systems with a strong emphasis on security issues[4]. However, the type of usage of cloud based services is not clearly addressed on the information available online. This system is deployed worldwide, including countries such as Belgium, Brazil, China, France, Italy, Netherlands, England and United States.

D. Siemens - eTicketing systems

The ticketing solution developed by Siemens is being used in the railway systems in Portugal, Belgium, as well as in Wiener Linien, Berliner Verkehrsbetriebe and Basler Verkehrsbetrieb, covering over 175000 users[5]. Ongoing projects will also deploy this system in Swiss Federal Railways and Swiss Association of Public Transport. As most of the solutions presented before, this platform supports multimodal systems under multiple operators. The information available online is however limited in which concerns the use of cloud based services.

One of the key security aspects is the fact that the platform aims to minimize the amount of data stored. Privacy is enhanced as the users are allowed to travel anonymously. Rate calculation can be done explicitly by the passenger in a hop-by-hop way, or end-to-end in an automatic fashion through the use of smartcards.

The Siemens platform comprises a central back office based on SAP for sales management. In addition, it provides mobile applications that the end-user can use to by tickets.

E. Rail Settlement Plan

Rail Settlement Plan (RSP) has developed a cloud based ticketing system for the English railway system[6]. The motivation for a cloud based approach was triggered by the need to satisfy requests in rush periods and also to reduce the infrastructure costs. This system is used in the following railway systems: Eurostar, Chiltern railways, East Coast, First Great Western, Virgin Trains, South West Trains and Scot Rail.

The RSP technical solution relies on Amazon Web Services (AWS), including Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Map Reduce (EMR). In addition the database and reporting servers are based on Infobright and Jaspersoft.

[2]http://www.acs-inc.com/public-transport/central-mgmt-systems/altas-public-transport-ticketing-system.aspx

[3]http://www.accenture.com/us-en/Pages/service-ticketing-new-generation-public-transport.aspx, http://www.accenture.com/us-en/Pages/insight-mobile-ticketing-public-transportation.aspx

[4]http://www.gemalto.com/press/Pages/news_1011.aspx

[5]http://www.mobility.siemens.com/mobility/global/en/complete-mobility/Pages/complete-mobility.aspx

[6]http://www.betterez.com/, http://www.computerweekly.com/news/2240172855/UK-rail-ticketing-services-get-next-generation-facelift-with-AWS-cloud, http://www.atoc.org/about-atoc/rail-settlement-plan, http://www.railway-technology.com/news/newssmart421-provide-live-sales-management-system-rail-settlement-plan

Security issues comprise different levels, from the security AWS mechanisms to access and identity management based on ForgeRock.

F. click1VIETNAM

The inter-city bus system in Vietnam relies on a cloud based ticketing system[7]. This system supports different bus operators, comprising functionalities for the bus operator, tourism agents and passengers. Bus operators can manage routes, employees, reservations, access control and financial reports. Tourism agents can access reservations and sales, while passengers can perform reservations, electronic payments and access their traveling history.

G. redBus

redBus is an Indican company that provides a Software-as-a-Service (SaaS) application to over 200 bus operators[8]. The redBus system relies on the following technologies: Amazon EC2, Elastic Load Balancing, Amazon RDS, Amazon S3, Amazon EBS, and Amazon CloudWatch.

Security and resiliency issues are supported by the native capabilities of AWS, including adaptive load balancing, redundant infrastructure and automatic recovery.

H. Additional Cloud Based Systems

This section presents could based solutions which are not specifically devoted to transportation systems and ticketing. The cases described aim to highlight the flexibility of cloud based systems, and their added value in the context of services supported on the Internet.

- TICKETLEAP: online ticketing service with mobile service support for shows.
- ERICSSON: solutions based on SaaS in areas such as telecommunications, intelligent transportation systems and mobile sales systems.
- GOL: South America airline that has developed onboard entertainment systems based on cloud services.
- FLAVOURS: online ticketing system for shows with the support of mobile services.

I. Link Consulting Solution

The ticketing solution developed by Link Consulting and Lisbon Polytechnic Institute [3] is based on the cloud and follows a SaaS approach. Therefore, different transport operators can subscribe this product. The end-system ticketing devices are integrated within the cloud following a thin client approach.

Cloud based services include data access, business, and business processing services. Data access is used in internal services that require information about clients, cards and sales. Business services are available on the cloud and support tasks such as the registration of new clients and sales authorization. Business processing services support the integration of the business services to provide the different system functionalities to the end-users, including card reading, ticket search, ticket selection, payment, card charging and sale registration.

On the client side, two possible approaches are possible, namely thin and fat clients. Thin clients have a limited set of functionalities, comprising the interface with the user and the communication of requests and replies with the cloud. Fat clients also support offline operations when Internet access is not available. When compared with traditional solutions, the integration of the clients in the cloud improves flexibility, scalability, availability and allows the support of heterogeneous clients. There are however security and privacy issues that require additional access control mechanisms.

J. Summary

The main characteristics of the systems analyzed in the previous sections are the following:

- Cloud based services.
- Resiliency mechanisms, alert triggering, automatic recovery upon failure.
- Booking, online sales, reporting.
- Security in communications and data storage.
- Privacy guarantees.
- Centralized management through a back office and applications for mobile devices.
- Dynamic definition of rates taking into account transportation and user type, date, rate plan, promotions, etc.
- End-to-end accounting in multi-modal systems based on routes, user and transportation operators involved.
- Remote configuration of devices and rates.
- Near Field Communication with the possibility to perform automatic detection of users in control points.

The analysis of the previous solutions has shown that while some of the ticketing solutions are standalone, others are integrated in intelligent transportation systems. It should also be highlighted that several of the analyzed systems use Amazon Web Services on the cloud, relying on its resiliency, security and scalability capabilities.

As a final remark, it has been noticed that most of the approaches used for the description of the solutions target potential clients, focusing on functionalities and with a very limited technical depth. Moreover, each of the products assessed is presented according to the profiles of such clients, which results in a rather heterogeneous source of information. Hence, we must say that the set of characteristics identified in this section provide only a limited input for our own work.

III. Architecture

A. The Intermodal Transport System of Oporto

Any ticketing solution of a reasonable size, comprising multiple companies with integrated sales must include an analysis system on top of the online processing system. Besides giving precious market information to the administrators, this analysis is necessary to divide the income among the carriers. Since clients may use a single transportation title in two different carriers, they may buy a title in one carrier that is valid in

[7]http://www.click1vietnam.com

[8]http://aws.amazon.com/pt/solutions/case-studies/redbus/

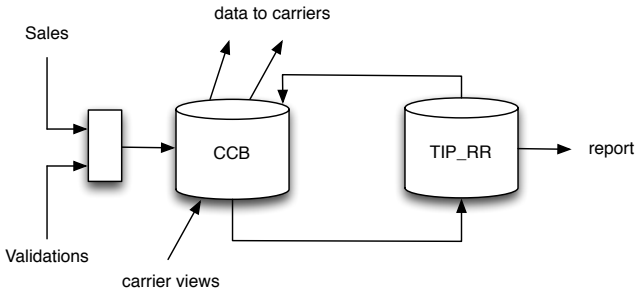

Fig. 1. TIP Ticketing System

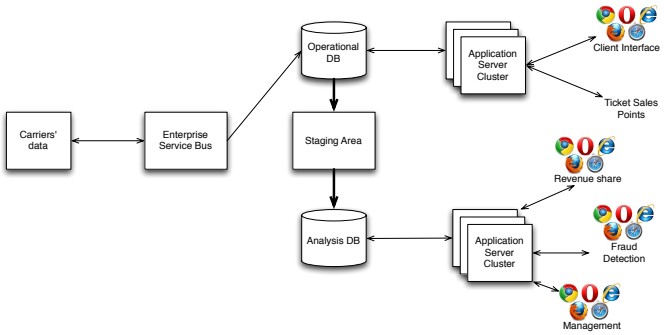

Fig. 2. Three-layer architecture

another. In the end, each carrier must get its own fair share of the income, based on data of the actual trip.

The importance of the analysis was eventually recognized on the Intermodal Transport System of Oporto (TIP). While the initial design of the system foresaw and used a single database (DB), as this database grew in size over time, the revenue sharing operation consumed an increasing amount of time. At some point, and according to the TIP system administrators, the evaluation of a month worth of revenues was taking 20 days, thus bringing the entire system close to an outright collapse. This unique DB was clearly overloaded, because it supported the analysis of validations concurrently with their usual actions in line, such as receiving sales data and validation, calculation of income from sales of tickets, update and query information to passengers with passes etc. To overcome this problem, the maintenance team ended up dividing the database in two parts, comprising an online Database (CCB) and an analysis database (TIP_RR), as we depict in Figure 1.

The system maintenance team progressively duplicated data onto this second database, which is a simple copy of the first one. This sliced the analysis time from 20 to only 2 days, a 10-fold improvement. Once it calculates the sharing of revenues, the system generates a report for the company managers. This architecture includes restricted data views, where carriers can see their own data, e.g., sales, but have no access to their peers contents.

The main conclusion we achieved by overviewing the running system with the Octal team was perhaps the need to separate the daily processing from the analysis database.

B. The New Architecture

Having a separate analysis database is one of the fundamental design options we considered for the architecture:

- A standard three-tier web-based interface.
- Support for all the existing hardware in buses, trains, ticket vending machines, etc.
- A separate analysis database.

Based on these constraints, we illustrate a high-level view of our system in Figure 2, with the presentation layer, which generates HTML for the browser, the application server (business layer) and the data layer, represented by the operational database. We also depict the analysis database, together with a staging area, which we shall discuss in Section IV. However, there is more to this system than the "three layers", because a significant and very important part concerns the connection to the ticketing hardware. This can be seen on the left side of the figure. The "carriers' data" rectangle represents the relevant sources of data, like validation or sales points.

This system thus have two important points of access to the database: one is the web server, the other is the transportation hardware through the Enterprise Service Bus (ESB). The utilization of the ESB provides a great flexibility on the number of systems it may support. Indeed, typical technologies, such as JBoss ESB [9] or Mule ESB [10] can access data in many different formats and using many different protocols. For example, they may read a binary file from a remote FTP server and apply the necessary transformation to the data, but they may also receive data in a client-driven push-based interaction using the HTTP protocol. The ESB is, therefore, at the heart of a smart connector architecture that transparently handles a large range of systems and equipments.

The operational database also interacts with the analysis database through a staging area. Besides revenue share, the analysis database (warehouse) can serve for other important tasks, like market analysis or fraud detection.

C. Utilization of the Cloud

A fundamental point of this project is to enable the utilization of the cloud to run as much as possible of the ticketing system. Following a very conservative approach, one may rent virtual machines in an IaaS approach, to remotely run pretty much any component of the architecture. This possibility might be particularly interesting for small carrier companies that want to off-load their own systems (which they use, for example, to collect data from buses).

A second, quite popular, possibility is to deploy the web server on the cloud, to transparently get the elasticity benefits from the cloud. This possibility is also very appealing because pretty much the same code might run on-premises or on the cloud, and, in the latter case, the web site might easily scale during load surges.

[9] http://jbossesb.jboss.org
[10] http://www.mulesoft.org

A final and more complex utilization of the cloud concerns the analysis part of the system. We devote the entire Section IV to discuss this topic.

IV. Data Management

A. Data Partition

One of the main problems in implementing a ticketing system and that has to be considered with particular care is the management of data. Without a proper partitioning of the data, it will not be possible to scale to large numbers of customers. The information generated by the ticketing and validation services is quite large and needs to be stored in a repository that must be efficient and capable of supporting fast computation. The central repository is a database. One of the tasks it supports is the revenue-sharing computation among the carriers involved in the transportation. It can also serve other objectives that should be added in the form of computations specified as executables for machines. From the point of view of the stakeholders, data are collected in a central repository that accumulates information over the useful life of that data.

From the central repository, the data are incrementally added to a twin repository, a data warehouse, with more or less transformations in between. "Data warehouse" is the usual nickname for a well structured repository of historical data for data analysis operations and computations in batch. Adding data into a data warehouse requires a set of more or less complex transformations of the source data, while a twin repository will be way less elaborate, similar to that found in Figure 1. Whatever the choice between the two options, the aim is that heavy computations, such as calculating revenue sharing, may be performed on this repository without weighing on the main repository, which serves operational applications. It is the classic question of processing decision support and calculations in batch, *versus* operational processing. Separation of concerns, with incremental data migration for decision support and calculations in "batch" is essential to avoid problems of efficiency in operational systems.

Managing the lifecycle of the data is important, since the accumulation of too much data over such a repository increases its size progressively. There are two possible options to appropriately limit the size of the data for the sake of efficiency. The first possible solution involves the identification of a period of validity for the data and passing data whose validity period has been exceeded to some archives. By keeping only fresh information in the master repository, a size suitable for efficiency is achieved. A second simpler solution uses the table partitioning mechanisms that some database engines offer. Instead of archiving old data, the idea is to do a partition of the data in the main repository into temporal intervals, specifying the creation of the main table partitioning by intervals of one or more temporal attributes. It will be necessary to choose a database engine that supports these mechanisms and that does so efficiently. Queries carried out on the system should specify a time interval, for example the last two years, and the database engine should be able to automatically do what is dubbed "prunning", consisting of accessing only the portion of data that is in the specified range, thus saving a considerable amount of processing. Computations that require access to all the data in addition to the recent one should be considered with care if the data size is very big, as some queries can take too much time.

In summary, there are solutions that can be used to ensure efficiency in the presence of voluminous data, requiring some care in the organization of data and computing.

B. Data Organization *in* Key-Value Stores and Cloud

When using cloud services, one must choose the type of engine for the database. While in Infrastructure-as-a-Service (IaaS) solutions, one can install the database engine he wants (or resort to those that are sold by the cloud vendor), in Platform-as-a-Service (PaaS) one will typically use integrated data management services on the platform. Usually, there will still exist the option of using relational databases with availability guarantees (ACID - atomicity, consistency, integrity, durability). In this case, it should be ensured that the server machine offers features appropriate to the amount of data and characteristics for the required query performance. Basically, one must ensure that the settings that a database administrator must take into account to ensure efficiency, are available in the cloud environment, and that the offered database engine can deal efficiently with the amounts of data that are predicted.

There is still another option, which can be used for some types of processing. Cloud *key-value* repositories are advertised as offering very good scalability and easy adding and removing of nodes, since reorganization involves only exchanging pairs of *key-values* automatically for load balancing. A simplified abstract view of these systems would be as an elastic *hashtable* or *HashMap*[11]. These systems do not attempt to replace traditional database engines, since they exchange ACID support for more flexibility and efficiency, and do not implement more complex computations such as aggregations or joins. The most basic primitive offered very efficiently on these systems are the *get* (search data indexed by the key specified by the user in the request), and the *put* (storage of data indexed by a key, indicated by the user). Such systems could be used for a number of features, especially for the calculation of revenue sharing by carriers. One might think about indexing the information given by validations using as key the transportation title identifier, adding validations to the corresponding title in a scalable manner. This data also grows considerably over time, particularly with a constant rate of addition of new tickets. Consequently, the computations also would increase considerably over time, once again being recommended the ability to archive older information, possibly on a twin scheme or other form of organization that may be wished.

[11] In the present context the word elasticity refers to the ability of a system to adapt to changes in computational load, making available more computer resources or removing them to the extent they are needed.

Fig. 3. All *on-premises*

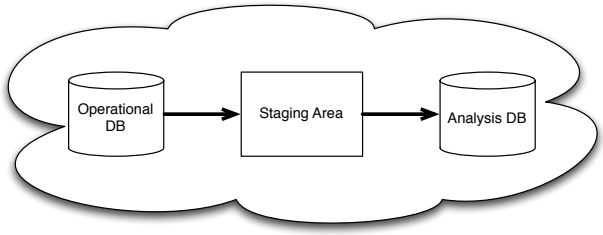

Fig. 4. All in the cloud

C. Data Movement

As mentioned, and also as used in the Oporto Intermodal Transportation system that was analyzed, it seems **essential** that the processes that require large amounts of data reside outside of the operational database. It will be totally impractical to lock access to a large amount of records in a particular database for calculating for instance revenue sharing, thus preventing or delaying normal daily routine transactions. The solution to this problem will normally go through a doubling of the data. Figure 2 already represents this, through the existence of a movement area (staging area), which serves to transport the data from the operational database to an analysis database.

The *staging* area allows one to apply a much more complex data transformation, inline with traditional ETL. For example, one can let the data grow for some time, say a week, such that validations of transport titles on all vehicles become available, or some faulty system recovers. Once the data is complete, one may then load all the data at once in a format compatible with the warehouse. The staging area also makes a preliminary analysis of the data possible. For example, one can group the data according to the company and compress it in order to minimize the information that follows for analysis. This possibility is particularly important if the analysis features reside in the cloud, as the cost of storage is typically proportional to the data size. There are other possible reasons for using a staging area. For example, it may be necessary or useful to use other existing data in the online database, to manipulate the data on the staging area. Linthicum [5] provides additional insight into this topic, pointing some more reasons, such as using less computing resources or networks to transfer data by parts instead of bulk. We also believe that using a batch treatment as opposed to a load dispersed over time is not particularly complex and may offer high performance.

D. Locating Architecture Components

With regard to the location of resources, there are economic and scalability considerations that should be taken into account. To do this analysis we consider the existence of four main components: the operational database, the analysis database, the staging area, and the data processing. Each of the four components may be *on-premises* or in the cloud. As some of the combinations do not make sense, in this section we focus on the possibilities that seem more important (for example, it would be difficult to understand a solution in which the staging area was placed in the cloud and everything else *on-premises* or the other way around). Also, we are not particularly interested in the case where everything is *on-premises* (Figure 3) or everything in the cloud (Figure 4). The first, because it involves no costs with the cloud, the second because it does not involve costs of data transfer, despite being relevant to our study. This latter case is also partially addressed in Section V.

In addition, the location of each component raises specific problems. For example, consider that the operational database resides in the cloud. This solution raises a potential problem of latency and costs of transferring and storing data. Regarding latency, we point to a study of games based in the cloud, by Chen *et al.* [1]. The authors of this study conclude that it is possible to play some games in the cloud, and given that few activities are so demanding from the point of view of interaction as the games, we think that the latency by itself, may not prevent resources from being on the cloud. Regarding storage, we discuss issues related to costs in Section V.

Keeping the operational database *on-premises* also raises questions, because it requires creating and maintaining all computer support that ensures performance and availability. In addition, there is a need to keep a copy of the data[12].

We will not consider the case where the online database runs in the cloud and the analysis database runs *on-premises*, for two important reasons: the data transfer tends to be cheaper (if not free) from outside to inside the cloud; the most prolonged computations take place over the analysis databases. Thus, we consider the cases where the analysis database resides in the cloud, including combinations in which the staging area can be inside or outside the cloud. We will not further analyze this last case, but it is easy to understand that, if this area stays out, there may be some savings in data storage, especially when there is a large data compression within the staging area, i.e., whenever it may be possible to create a much smaller summary of the original data size.

Consider now that the operational database is *on-premises* (Figures 5 and 6). If the analysis database is in the cloud, we must consider, in addition to the cost of storing data and transfers, the appropriate choice of the database technology (relational vs. *key-value*). We must also take into account characteristics scalability, to make it possible to execute the necessary calculations, e.g. revenue sharing, efficiently. An

[12]One should note that there are cases in which some clouds are known to have lost data including clouds of well known suppliers, hence we should change our perspective and put the question as follows: when the data is controlled by the customer company, it can know exactly the risks involved; when consigned to a cloud provider, it becomes subject to a *Service Level Agreement*, which by itself will not allow data recovery, if the cloud vendor makes a serious mistake or has an accident with the data.

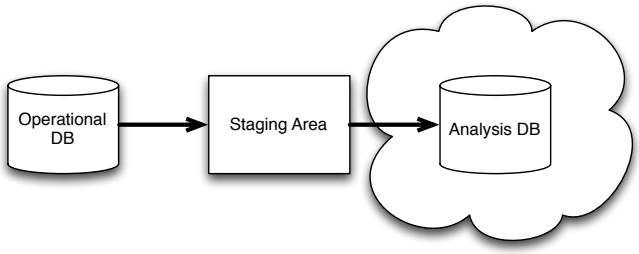

Fig. 5. Staging area *on-premises*

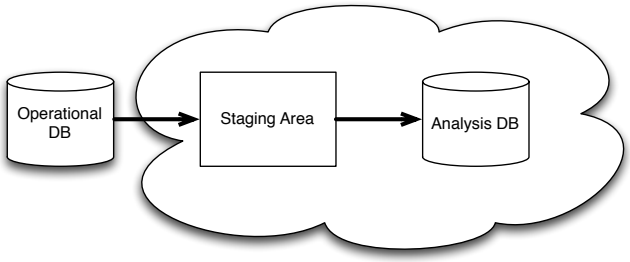

Fig. 6. staging area in the cloud

important factor, usually pointed in favor of the cloud, is the lower *Total Cost of Ownership*, although the truth of this statement is questionable and might depend on the particular cases.

The location of processing is also important, although it this can never be completely orthogonal to the location of the data. If data are stored in the cloud, it hardly makes sense to process them *on-premises* and vice versa. The question that arises is to know how one can achieve automatic elasticity for an application. One possibility is to use a PaaS approach with vendor-specific APIs (e.g., Google Application Engine). This approach has, however, two disadvantages: the use of an API of this kind may require a very specific set of skills not very common among IT professionals available in the market; secondly, it creates *lock-in* to the seller adopted, because it becomes difficult to switch to another cloud provider or to an *on-premises* solution *afterwards*.

On the other hand, the great advantage of the IaaS approach is to not involve any *lock-in*, being very flexible. The shortcoming of this option is that elasticity is usually not automatic in this case.

V. Cost Analysis

In this section, we focus on studying the different architecture alternatives with regards to the cost of using the cloud to deploy the ticketing system.

A. Introduction

Placing a ticketing service in the cloud, with the goal of fulfilling operational requirements as well as analytical processing, is in itself a challenge. In the present case, that challenge is subject to a technical constraint which requires the ticketing service to be provided by means of a *public cloud* or a *private cloud*, depending on customer preference.

On the one hand, we consider a scenario in which the transport operator acquires only terminal devices (such as vending terminals) that connect directly to a public cloud. On the other hand, some transport operators are interested in administrating the entire system, becoming responsible for managing the infrastructure in a private cloud (that is, a local datacenter).

In between the two extremes we consider intermediate options, in which the ticketing service is divided among a private cloud and a public cloud. Such a *hybrid cloud* scenario allows one to deal with sporadic heavy tasks (allocated to a public cloud capable of offering a greater elasticity) and maintain some features and sensitive data in private datacenters (thereby respecting confidentiality requirements).

B. Existing Cloud Solutions

There are numerous public cloud operators in the market, as well as several solutions for creating private clouds. However, the possibility of creating a single implementation capable of functioning in a public cloud as well as in a private datacenter is intrinsically dependent on the *service model* adopted.

According to the PaaS (Platform as a Service) model, the cloud consists of a service with a given operating system, storage and data management solutions, fundamental services, and one or a few frameworks for application execution. Applications are developed and placed in the cloud without the need to build and maintain the underlying platform. The PaaS model is adopted by several cloud providers, such as Microsoft Azure, Google App Engine, JoyentCloud and SoftLayer (IBM).

Another service model that offers interesting possibilities for the ticketing service is the IaaS (Infrastructure as a Service) model. Under this model, a cloud operator provides virtual machines in which a customer may install different operating systems (with some limitations), as well as the applications. In this model, customers manage the service much like physical machines in a remote datacenter. Some examples of IaaS providers are Microsoft Azure Virtual Machines, Amazon EC2, JoyentCloud, HP Cloud, Rackspace Cloud and Google Compute Engine.

C. The Ticketing Service

Regarding the ticketing service, the IaaS differs the PaaS model in three key aspects: the flexibility of the software development process, vendor lock-in, and total cost of ownership.

The first aspect is the flexibility offered to the software development team. In the IaaS model, the software (including the operating systems) are managed by that team. There are, consequently, less technical restrictions limiting the development. This possibility offers the ability to reuse software components. However, this flexibility leads, in the long run, to a higher operational cost.

The second aspect is vendor lock-in. Regarding this issue, it is more favorable to use IaaS, as the development team may

choose (practically without limitations) any operating systems, platforms for executing applications, databases, etc. Unlike the IaaS model, the PaaS model leads to a greater vendor lock-in associated to the choice of cloud operator and respective components belonging to the software stack.

The third aspect is the total cost of ownership associated to choosing among PaaS and IaaS, which includes the development cost and the operational costs. The operational costs break down into a monthly price established by the cloud operator, and the cost of maintaining and administrating the application. With this in mind, we focus on the cost of using the cloud for the ticketing service.

D. Cost Simulation

In spite of the complexity of modelling the cost of running an application under the PaaS model (the computational needs are not always simple to estimate) it is possible to make a fair comparison using the Microsoft and Amazon cost simulation tools.

We begin by comparing two installations (a small setup and a medium setup) with regards to the monthly prices under the PaaS and the IaaS models. The two setups are characterised as follows:

- *Small infrastructure.* This installation would be appropriate for a small transport operator (for example, anywhere between 5 and 10 vehicles). We assume that the operator manages only terminal equipment and, consequently, the remainder of the infrastructure is in the cloud. We consider that the computational needs demanded by this system correspond to 2 virtual cores at 1.6 GHz with 4 GB of memory and 500 GB of storage, using 100 GB in monthly outbound network traffic.
- *Medium infrastructure.* This installation corresponds to the needs of a transport operator for a city with 100 000 habitants. In this installation we equally assume that the infrastructure is entirely placed in the cloud. We consider that the computational needs to run this system corresponds to 4 virtual cores at 1.6 GHz, 8 GB of memory and 1 TB of storage, with 1 TB in monthly outbound network traffic.

Comparing the two setups (small and medium) with regards to their cost under the IaaS model and the PaaS model, we obtain the results in Table I.

TABLE I
PRICE COMPARISON FOR SIMILAR CONFIGURATIONS UNDER THE PLATFORM AS A SERVICE MODEL AND INFRASTRUCTURE AS A SERVICE MODEL.

	Small Size		Medium Size	
	IaaS	PaaS	IaaS	PaaS
Computation	100	89	199	177
Storage	27	-	59	-
Network	9	9	91	91
Database	-	94	-	132
Total	136 €/mês	192 €/mês	349 €/mês	400 €/mês

The prices of the PaaS versions correspond to the described installations simulated in the Microsoft Azure tool. The IaaS versions correspond to Microsoft Azure as well, but the Amazon EC2 prices were equally analysed, without any significant differences between the two providers. Therefore, the monthly price for renting virtual machines (in the IaaS version) is representative of the market. It should be noted that, in the PaaS model, the storage cost is included in the computation cost, and that in the IaaS model the database is managed along with the remainder of the storage.

We can observe that the small differences between IaaS and PaaS are owed essentially to the pre-installed database managed by the cloud operator. Given that the database will certainly require maintenance, the IaaS and the PaaS models are similar regarding the monthly prices. The reason for this is that, in the IaaS model here analysed, the database is managed by the team responsible for the ticketing service (rather than the cloud provider).

With regards to computation, the table was constructed assuming a usage close to 100% of the resources in the PaaS model. This assumption may be too high, since PaaS applications tend to benefit from optimisations carried out by the cloud provider. Taking this into account, the PaaS model may actually be the most economical.

In any case, it is important to notice that the monthly price to pay the cloud operator may be relatively low when compared to the total cost, including development, maintenance, and operation of the ticketing service. In the development phase, the IaaS has a lower cost, since the development team can maintain the same technologies with which it is used to. In the PaaS model there are specific APIs that have a learning curve, and that may lead to a greater development cost and cause delays, depending on the chosen platform.

However, this initially higher cost of the PaaS model is amortised during maintenance and operation, since the cloud operator is responsible for a significant part of the tasks. Many of the operational costs are therefore transferred to the cloud operation, who divides them among a great number of customers. Therefore, when considering the total cost of an application, in the long term, the PaaS model is the most attractive.

Having compared the costs of two installations (small and medium) in the IaaS model and in the PaaS model, it is now important to analyse the costs of a very large installation. We consider such an installation in a public cloud and in a hybrid cloud:

- *Very large installation in a public cloud.* This installation would be appropriate for a very large transport operator (for example, in a very large city). We assume that the transport operator choses to deploy most services in a public cloud. We consider that the computing system requires 24 virtual cores at 1.6GHz, 150GB of memory and 6TB of storage, and requires 2TB in monthly outbound network traffic.
- *Very large installation in a hybrid cloud.* The size of the installation is the same as the previous one, for a very large transport operator. Unlike the previous scenario, the

installation is carried out in a hybrid cloud model, in which the on-line processing components are managed by the transport operator, but the analytical database (concerning sporadic processing, reporting, and accounting) is placed in a public cloud. The requirements for handling the sporadic processing consist of 8 virtual cores at 1.6GHz, 50GB of memory and 2TB of storage. We consider that this configuration executes 2 days in each month, to compute reports. There is very little outbound traffic, since the vast majority of the network traffic corresponds to loading data into the cloud operator's network. We therefore consider only 50GB of outbound traffic and that inbound traffic is not charged by the operator.

TABLE II
MONTHLY PRICE COMPARISON FOR A VERY LARGE TRANSPORT OPERATOR IN A PUBLIC CLOUD AND HYBRID CLOUD SETTING.

	Very Large Operator	
	Public Cloud	Hybrid Cloud
Computation and Storage	2344	50
Traffic	178	4
Database	1178	393
Total	3700 €/mês	447 €/mês

From Table II we conclude that the monthly costs may be reduced by using the cloud's elasticity to execute sporadically heavy tasks. One should note that the table only shows the cloud operator's costs. In a hybrid cloud, only sporadic tasks are executed in the cloud. Assuming a usage of 2 days per month, the price involves only those 2 days. The network traffic, accordingly, has a very low cost in both cases, and is almost null in a hybrid cloud setting.

E. IaaS and PaaS in a Private Cloud

Given that one naturally prefers to avoid vendor lock-in and devise an architecture for the ticketing service that allows public and private cloud deployment (as well as hybrid options), the flexibility of the IaaS model, in which an application is deployed in remote virtual machines, is the most interesting.

In fact, this flexibility makes the IaaS the most adequate model for the ticketing service. However, there are some important alternatives and indications that such alternatives may grow in the near future. The Microsoft Azure platform, for instance, accepts applications compatible with Windows Server (written in .NET), through Microsoft System Center and Hyper-V Server.

Therefore, it is possible to install a private cloud only using Microsoft technologies that, not being Azure itself, assure compatibility for the applications. This option has very high costs and demands a very solid knowledge from the development team, regarding the platforms and virtualisation solutions by Microsoft. Given that such costs may be unbearable, the ideal solution would be to base a solution on Azure for private clouds. At the present moment, such a solution does not exist, although Microsoft has announced that it would make it possible to buy servers with Microsoft Azure pre-installed. Manufacturers such as Fujitsu and HP have confirmed these initiatives, but to our knowledge there are no concrete market offers.

Regarding Google App Engine, it may be viable to convert a public cloud in a private cloud, through AppScale. This initiative aims to provide, in open source code, the same APIs offered by the Google App Engine, and allows one to install in a private cloud the same applications that run in a public cloud. Several public cloud operators mentioned in this document have announced private and hybrid cloud solutions, being therefore reasonable so expect that such solutions will be more frequent in the near future.

F. Final Cost Evaluation

The PaaS model has, in a general way, a total cost lower than IaaS. We can observe that the small differences between IaaS and PaaS are owed essentially to the inexistent of a database pre-installed and managed by the cloud operator. Although the monthly price is lower with IaaS, the operational cost (including maintenance and operation) dominates and should be lower. In other words, the monthly price is unlikely to make a difference when compared to the operation and maintenance costs.

To our knowledge there is no private cloud solution for running Microsoft Azure, although several announcements lead us to believe that this may change in the near future, with the presentation of an Azure solution for private datacenters. It is currently possible to configure Windows Server in a private cloud, maintaining compatibility with Azure applications, but the costs of such an approach can be expected to be very high.

Google's App Engine has an open source clone that allows one to run the same applications in a private cloud as in the public cloud. There are also other cloud operators offering similar solutions, and one should expect solutions supporting private clouds to be more common in the future.

In any case, only an IaaS solution may avoid the vendor lock-in effect at the technology level. This may often be a less important matter, as in general companies chose to be dependent on some vendor's technology, and admit the vendor lock-in from the strategic point of view.

Finally, one may observe a return on investment associated to solutions using hybrid clouds, whenever a portion of the processing is sporadic. If such processing is allocated in the cloud, and corresponds to a small fraction of the total processing, the monthly cost reduction is likely to become attractive.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we presented a typical three-tier architecture for a ticketing service. Central to this effort was the evaluation of the options and costs to run all or part of this system on the cloud. One of the most important aspects that we pointed out in this document and that is reflected in the solution adopted by the Intermodal Transportation System of Oporto (TIP) is the need to separate the online and the analysis

databases. This latter database may not even be relational, if we want to take advantage of parallelism, to exploit the elasticity of cloud resources. Ensuring this parallelism might be one of the biggest challenges ahead given the option for IaaS clouds, possibly made of public and private resources, in a hybrid configuration. The preference of Octal falls on an Infrastructure-as-a-Service cloud, particularly in the case where we consider a hybrid cloud.

In particular it will be crucial to have a solution that can use more resources from the cloud, whenever that is desirable in view of the cost/performance ratio. Small operators may resort entirely to the cloud to free themselves from operational costs. On the contrary, large operators may rather prefer the cloud to accommodate sporadic loads, particularly over their analysis database. This possibility will lead to savings in operational costs, but it is necessary to assess whether the increased risk of developing a hybrid solution brings any savings on a longer term.

This paper still leaves a number of important open points that we need to address as future work. The most interesting one is how to create an analysis database with transportation data that allows parallel execution of queries that can further accelerate tasks such as sharing the revenues.

REFERENCES

[1] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 1269–1272, New York, NY, USA, 2011. ACM.

[2] Jeri Edwards. *3-Tier Server/Client at Work, Revised Edition*. Wiley, February 1999. Revised Edition.

[3] João Ferreira, Porfírio Filipe, Gonçalo Cunha, and João Silva. Cloud terminals for ticketing systems. In *The Fifth International Conferences on Advanced Service Computing*, 2013.

[4] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard), December 2002. Updated by RFCs 5343, 5590.

[5] David S. Linthicum. Best practices in leveraging a staging area for saas-to-enterprise integration. http://www.informaticacloud.com/images/whitepapers/WP-Leveraging_a_Staging_Server.pdf, 6 2009.