Cloud.Jus: Architecture for Provisioning Infrastructure as a Service in the Government Sector

Klayton Castro, Gabriel R. D. Macedo, Aleteia P. F. Araujo and Leonardo Reboucas de Carvalho
Department of Computer Science, University of Brasilia, Brasilia, DF, Brazil

Keywords: Infrastructure as a Service, Interoperability, Community Cloud, Multi-cloud, Federation.

Abstract: Building a community cloud by federating private clouds is one of the lower cost alternatives for hosting applications that require distributed deployment to meet scale-saving, high availability, reliability, and service level compliance. Despite its potential benefits, there are many issues about lack of standardization, system integration, interoperability and portability across multiple service providers, resulting in low adherence to the model in organizations that are still struggling to adapt its legacy applications to a cloud architecture in complex environments, such as some governmental sector scenarios. Currently, there is no seamless approach to migrate from the traditional infrastructure model to a cloud computing model on these organizations. So, we propose an architecture for building a community cloud even in scenarios with a strong presence of non-cloud native applications by developing a low-coupled infrastructure middleware that supports different hypervisors, a GUI and a CLI. To show the feasibility of our approach, we evaluate the architecture on a set of infrastructures at Superior Courts of Brazilian Judicial Branch, that may compose a cost-effective solution to start the transition to the cloud model in other organizations also.

1 INTRODUCTION

Part of information technology (IT) areas in organizations focus its controls on their work processes, resulting in a culture oriented to technological silos, particularly in the infrastructure provisioning, becoming its environments quite complex, which require ever-increasing and specialized management teams (Sosinsky, 2010). Besides, a not suitable infrastructure creates an uncomfortable situation for administrators, who need to support the growing demands very quickly while maintaining the environment stability (Ahn et al., 2014). This way, to build integrated IT solutions is a meaningful challenge, because it is hard to create universal service platforms in environments that result from disaggregated business rules for each system (Edwards, 2012).

Then, the cloud computing model stands out, quickly demonstrating its ability to transform the path of managing IT resources, making them more agile and efficient (Kim et al., 2016) by providing on-demand access as a service to a shared set of configurable assets, such as network, servers, storage, and applications, that may be dynamically provisioned and instantiated with minimal effort (Buyya et al., 2013). Although the cloud computing model has become a favorite choice about a decade ago, establishing itself as a reality for many organizations, there is still a lack of standardization and uncertainties about interoperability, integration, and portability over multiple infrastructures (Baur et al., 2015).

Also, the literature recommends to employ different infrastructures, both to ensure the IT services availability and to avoid the vendor lock-in, that is the blocking effect on a single provider, aiming to avoid risks concerning reduced responsiveness, resilience, and service continuity when relying on a single on-premises datacenter (Petcu et al., 2013). To overcome this high-coupling hurdle, virtualizing all applications and service dependencies, associating them with cloud middleware and standardized interfaces is an option (Brasileiro et al., 2016a).

However, each provider applies its standardization patterns, and there is enough room to improve the interoperability issues, such as storage access methods, virtual machine (VM) image formats, application and services resources provisioning through Application Programming Interfaces (API) (Bégin et al., 2008). Besides, there are some open programming libraries able to deal with some virtualization vendors and Cloud Service Providers (CSPs), such as libvirt, jclouds, and libcloud (Wickboldt et al., 2014), that

Castro, K., Macedo, G., Araujo, A. and de Carvalho, L.
Cloud.Jus: Architecture for Provisioning Infrastructure as a Service in the Government Sector.
DOI: 10.5220/0007731804120421
In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)*, pages 412-421
ISBN: 978-989-758-365-0

also requires integration into preexisting interfaces to become productive. There are some initiatives aimed at making general specifications useful for APIs, such as the Open Cloud Computing Interface (OCCI) and Cloud Infrastructure Management Interface (CIMI), which are still not very mature and evolve in a rather timid manner, whose CSPs demonstrate a low interest to support them fully (Baur and Domaschka, 2016). Other approaches accommodate similarities to CSP's industry, giving rise to open Cloud Management Platforms (CMP), which propose resources orchestration in a cloud manner (Baur et al., 2015). Nevertheless, the most prominent CMPs promise very close features and capabilities, making it difficult to select the most appropriate tool for each organization. Also, such tools carry a certain complexity of configuration and do not instantly provide as many resources as desired and necessary, not rarely leading to exhaustive efforts to promote satisfying deployment (Wickboldt et al., 2014).

This way, migrating an existing application or building a new one using a cloud-native architecture requires a lot of research and development efforts to take advantage of available technologies and mitigate the rework risks, that is not whenever possible. Thus, our initial motivation is to reconcile the use of IT infrastructure assets in the Brazilian government sector, especially at Superior Courts of Judicial Branch of the Union (PJU), modernizing them to a cloud-based architecture. Similarly, we believe another set of partner institutions can collaboratively share their infrastructure in the same way, supporting the investments' assertiveness, and avoiding too many operational costs when dealing with complex environments.

In general, medium-sized organizations that already implemented on-premises data centers observe low use of the investments made in counterpart to the present potential. Besides, many institutions such as Brazilian Superior Courts have available a lot of not shared IT infrastructure resources: secure data centers, reliable communications networks, large storage platforms, and modern server equipment, with leading market virtualization systems, i.e., VMware (VMware, 2018), Hyper-V (Microsoft, 2016) and Oracle VM (Oracle, 2018). Such IT professionals and geographically distributed robust assets are available but still isolated on each organization.

Then, it is natural that create a community cloud by federating these resources offer advantages over pure private clouds. Following this approach, if at least a portion of the infrastructure present in each organization, it is possible to develop a typical and seamless architecture, regarding the particularities of each environment. So, the question arises: is it possible to define an Infrastructure as a Service architecture to abstract the underlying complexity, based on low-coupled design and non-intrusive standardization patterns, able to enhance the benefits of the cloud computing model at governmental organizations?

2 STANDARDIZATION AND SERVICE ORIENTATION CONCEPTS

Performing cloud computing consists in an infrastructure agnostic way for deploying multiple placement independent resources, maintaining its transparency (Vaquero et al., 2008). Thus, the distributed services and the data exchange uniformity mechanisms contributes significantly for developing efficient applications according to this paradigm. The improved virtualization technologies associated with the application of Service Oriented Architecture (SOA) become widely useful (Buyya, 2009).

Concerning an abstraction of these services, there is a cloud management layer, which consists in: (1) API (Application Programming Interface), (2) CLI (Command Line Interface), (3) GUI (Graphic Interface User Interface), in general called Dashboard, and (4) Orchestrator components, which are responsible for performing essential functions such as instantiating, deleting and managing VMs, monitoring usage, and power management (Dukarić and Jurič, 2013). In this way, APIs are sets of routines and protocols that act to intermediate the connection between two systems (operating system, web site) or specific services that talk to each other to perform tasks and process requests from users, as well as other independent systems or services (Paul et al., 2014).

Due to the interoperability needs in a low-coupled way, approaches based on the use of APIs are gaining prominence, since they are more transparent for the customers (Buyya et al., 2013). So, main cloud components are controlled by SOA mechanisms, maintained by communicating APIs, that supports the interaction with the underlying infrastructure, integrating completely different environments and components. Then, SOA implementation make possible to do fast interactions, focusing on orchestrator-based approaches, that establish service interfaces to the resources provisioning.

Table 1: Distinguishes among CMPs.

	OpenStack	OpenNebula	CloudStack	Eucalyptus
Supported APIs	AWS/OCCI	AWS/OCCI	AWS	AWS
Supported Hypervisors	Xen, KVM, VMware	Xen, KVM, VMware	KVM, Xen, VMware, Oracle VM	Xen, KVM, VMware
Architecture	Component Based	Low-Coupled	Strong-Coupled	Strong-Coupled
Hybrid Cloud	No	Yes	Yes	Yes

3 CLOUD MANAGEMENT PLATFORMS

Cloud computing model requires supporting large-scale elastic data centers, orchestrated according to an agile infrastructure model (Edwards, 2014). By operating in this way, the resources association becomes increasingly active, providing efficiency gains by assigning layers of abstraction that do not demand the user to know about the placement and other configuration details (Foster et al., 2008). In this scenario, free software-based cloud management platforms (CMP) are available, such as OpenNebula (Sotomayor et al., 2009) and OpenStack (Sefraoui et al., 2012).

According to Roveda et al. (Roveda et al., 2016), OpenStack stands out for corporate acceptance, although it is more fragmented, complex and robust, while OpenNebula offers simplicity and efficiency. While OpenNebula aims to manage private, hybrid or public clouds, according to the IaaS model, OpenStack fulfills covers all the infrastructure resource layers through a modular architecture, based on a series of interoperable projects. However, many companies, public organizations, and other institutions still do not take full advantage of these solutions, partly because of the uncertainties about using community-based software without commercial support or even lack of accurate information on such possibilities (Fiore et al., 2015).

In addition to OpenStack and OpenNebula, there are other useful CMPs, such as CloudStack and Eucalyptus, demonstrating that it is possible to structure such solutions efficiently, making them viable alternatives over commercial providers (Casaj et al., 2014). Lynn et al. (Lynn et al., 2015) and Parmar and Champaneria (Parmar and Champaneria, 2014) present a qualitative review of the most prominent open source CMPs in IaaS research field (OpenStack, OpenNebula, CloudStack, and Eucalyptus), whose main distinctions are in the Table 1.

Llorente (Llorente, 2013) states that presented solutions follow two philosophies of design, due to functional closeness of commercial implementations considered as state of the art: Datacenter Virtualization (OpenNebula and CloudStack, conceptually closer to the characteristics of VMware vCloud) and Infrastructure Provisioning (OpenStack and Eucalyptus, conceptually closer to the characteristics of Amazon AWS) as described in the Table 2.

Table 2: Different philosophies for IaaS.

	Datacenter Virtualization	**Infrastructure Provisioning**
Application Type	Multilayer, conventional defined	cloud native
Interfaces	API and Admin Dashboard feature-rich	Simple API and self-service portal
Management	Fulfit the life cycle of resources	Simple, with full abstraction of underlying infraestructure
Deployment Model	Private, mostly	Public, mostly
Design Approach	Bottom-up, defined by data center complexity	Top-down, definided by efficient cloud interfaces
Business Requirements	High-Availability, Fault-Tolerance, Replication and scheduling cloud promoted	Design depends on application
Data center Integration	Ease to adapt and get gain CapEx	Built on homogeneus market infrastructure.

4 INTEROPERABILITY APPROACHES

Cloud interoperability problems become more evident as CSPs deliver new products and features, while more companies migrate their applications to these environments (Ciuffoletti, 2014). It is necessary to enable a broader cloud ecosystem to take advantage of familiar to the market elasticity and service measurement into three-dimensional aspects (Petcu et al., 2013): (1) federation and communication policies between CSPs, (2) execution environment and migration support, and (3) abstraction of programmatic differences between CSPs in the architecture design.

When provisioning on-demand resources through cloud computing services, it requires appropriate communication interfaces between the existing abstraction layers to deal uniformly with each infrastructure segment (Demchenko et al., 2012). Aiming at representing the CMP structures generically, Dukaric and Juric proposed a taxonomy, that helps to create a systematic approach about similarity of terms, notions of semantics and relationships involved, considering most prominent CMPs features (Dukarić and Jurič, 2013), as described on following layers:

1. Resource abstraction: comprises the basic infrastructure (computing, storage, and network).
2. Main Services: Displays the highlighted components to enable resource delivery (identity management, scheduling of tasks, a repository of VM images, billing and event logging).
3. Support: it acts as middleware, providing communication mechanisms for cooperation and interaction with other components (message bus, database, and file transfer).
4. Security: includes authentication, authorization, and security groups).
5. Management: includes federation, elasticity, resources, users and groups, service levels, reporting, monitoring, incidents, power management, and resource allocation.
6. Control: Provides mechanisms for application and monitoring of service levels, measurement and policy definition, orchestration (automation of tasks such as creating, monitoring, updating, and deleting instances of virtual machines through API or CLI) and notification of events.
7. Value-added services: It complements the core services layer (establishment of availability zones, support for high availability, hybrid cloud support, live migration, among others).

5 RELATED WORK

Minimal management tasks on individual components can compromise applications for reasons that are not obvious to non-specialists. So, search for uniform management standard assists in proposing common interoperability mechanisms, meeting the need to combine resources and different suppliers, because of its heterogeneity and unknown dependencies. Although, universal processes are not entirely defined yet for the automated provisioning of topologies in complex sets of applications and their adaptation at runtime. So, it is essential to follow in an open approach, designed to be extensible. The literature focuses its efforts to standardize interfaces, to develop provisioning and cloud federation middlewares, and some hybrid approaches.

One of the most prominent standards used in more recent work is the OCCI (Open Cloud Computing Interface) (OCCI, 2018) that defines a REST API and a metamodel for IaaS environments. The ability to simplify the developer's work with cloud systems integration is one of the critical benefits of OCCI, which abstracts the differences between service implementations independently of the vendor (Korte et al., 2018). The use of middlewares is particularly useful, both to enable a resource-friendly transition of resources between vendors, and to promote the required integrations for the legacy applications. The hybrid approaches usually implement open source CMPs on dedicated architectures, compatible with existing standards (Toosi et al., 2014).

This way, compatibility with familiar CMPs is particularly useful for infrastructure provisioning in federated clouds composition, such as the Federated Cloud Platform (FCP), developed by EGI (European Grid Infrastructure), an association of academic institutions focused on high-performance computing. The FCP groups 37 institutional-oriented scientific clouds and gathers 21 providers running platforms such as OpenStack, OpenNebula, and CloudStack, as demonstrated in the works of Sustr et al. (Šustr et al., 2016) and Casaj et al. (Casaj et al., 2014).

Aiming to support the OCCI standard, emerge some extensions to the significant CMPs, presenting integrations with specific architectures, that can abstract the IaaS layer complexity following the OCCI patterns (Parpaillon et al., 2015). Par (Par et al., 2016) presents the rOCCI framework, a stack of components for creating client/server applications conforming to the OCCI standard. The rOCCI-core library implements the OCCI class structure, enabling it to be used natively in programming, while rOCCI-API implement OCCI transport services through the HTTP protocol. ROCCI-server relies on specific backends to communicate with various types of CMPs structures, acting as a kind of stateless translator. ROCCI-CLI is a command-line client for controlling IaaS compatible clouds, i.e., the OpenNebula.

However, Wickboldt et al. (Wickboldt et al., 2014) point out that a common design aspect of current CMPs concerns the black box-like control nature in which cloud administrators have little opportunity to influence the resources management, i.e., VM host positioning and virtual link path selection. The authors present the Aurora Cloud Manager (Aurora CM), whose architecture makes possible to control some aspects of infrastructure, such as the network flow traffic in real time, through interaction with the

API. The Aurora CM establishes an object-oriented API to make IaaS provisioning and resource management more flexible and, also, natively interacts with user-friendly external systems for improving the cloud monitoring (Cunha Rodrigues et al., 2016).

The deployment of applications that need to operate in different environments with identity management coming from multiple networks is addressed in the CYCLONE (Slawik et al., 2017) project. It is a middleware stack that allows you to deploy and manage multi-cloud applications and CMPs, including the implementation of identity federation, and a network manager that connects VMs regardless of the underlying infrastructure and use-case presentation in the area of bioinformatics. CYCLONE enables the federation of heterogeneous environments but does not offer a complete provisioning suite, involving features desirable to customers, such as monitoring and custom usage reports.

Similarly, FogBow is the middleware used in EU-Brazil's Cloud Connect project to federate private clouds hosted by Brazilian and European research institutions, running on the IaaS CloudStack, OpenStack and OpenNebula platforms (Brasileiro et al., 2016b). It bases on a higher architecture level on top of the IaaS cloud orchestrators of each member of the federation, and it presents excellent flexibility by employing dedicated plugins, which allow the definition of precise points of interaction between the federation middleware and the underlying cloud orchestrator. Like the previously described works, FogBow provides a standard implementation of the OCCI interface, enabling customers to interact with specific managers.

6 Cloud.Jus ARCHITECTURE

Despite serving users over all the country, there is an eager high-coupling effect in the Brazilian PJU, whose applications operates exclusively on its Courts at on-premises infrastructures. This scenario, similar to the vendor lock-in ones, sometimes threat the IT services availability due to maintenance on datacenters among other needs. Considering organizations must focus on speed, not completeness, and should think of tools in a chain approach, as this is the only design pattern that everyone can agree to standardize in complex environments, it is essential to ensure the low coupling sought as good architecture practice (Edwards, 2012).

In general, having individually right tools is the simplest and easiest way to accomplish something, and the most critical concern is how to integrate a needed toolset and how to manage logically and consistently. Thus, it is not advisable to create universal projects that wish to remedy all kinds of situations immediately. On the contrary, interchangeable components must be designed considering that partitioning reduces overall complexity and iteration increase the likelihood of success, highlighting the importance of low coupling systems (Sessions, 2006).

Large companies such Microsoft, Oracle, and VMware invest in the development of new resources and, as a result, these resources work best by using a combination of products native to these companies, called stacks (Llorente, 2013). So, it makes no sense to develop new tools that are already very mature and widely available in enterprise hypervisors. It thickens the possibilities for organizations to move more quickly to the cloud model since the legacy cannot be easily incorporated and there are requirements for adapting processes and applications, not suitable to avoid the vendor lock-in effect. Thus, our approach takes advantage of existing resources and native tools, focusing on the integration services and interfaces that make calls in a cloud way.

6.1 Design of Integration Standard

Categorization of services, interfaces, and roles involved must be well defined to evolve the development of the architecture. Any changes or exceptions that may break the agreements should have a predefined treatment, so a component, service, or process should be as tolerant of vendor switching as possible (Jamsa, 2012). A formal framework is not yet fully implemented for the emerging OCCI standard. However, the associated resource-oriented meta-model defines it as the best possible candidate to develop a set of tools to manage all types of cloud infrastructure resources as a service. The proposed architecture interacts directly with each corporate hypervisor CLI and API using native vendor tools in a comprehensible way at a lower abstraction level. So, it does not depend exclusively on third-parties libraries such as libvirt, that becomes a single point of coupling, nor the need to adapt local management in function of tools such as OpenStack, which require administrators to change how to deal with the legacy environment, making it a hard transition to the cloud model. Neither does depend exclusively on OCCI integrations, which still requires extensions. We employ the OCCI standard to build the messages used by IaaS platform services and developed a low-coupled middleware that translates these messages to each supported set of hypervisors.

For example, our low-coupled approach makes it

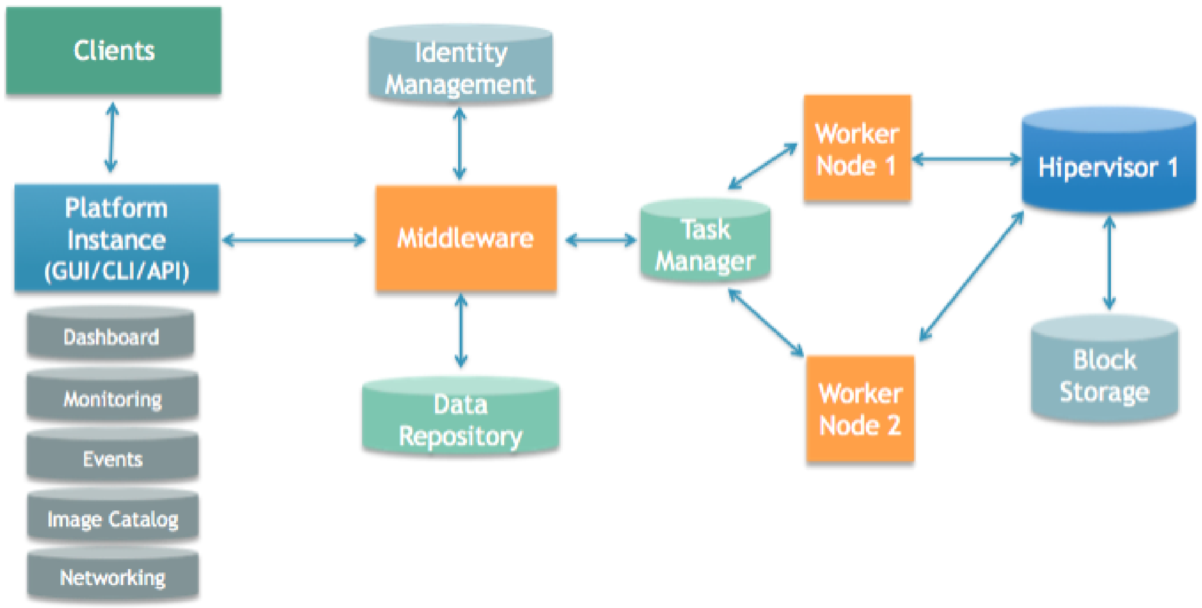

Figure 1: Interaction between middleware and orchestrator and resources.

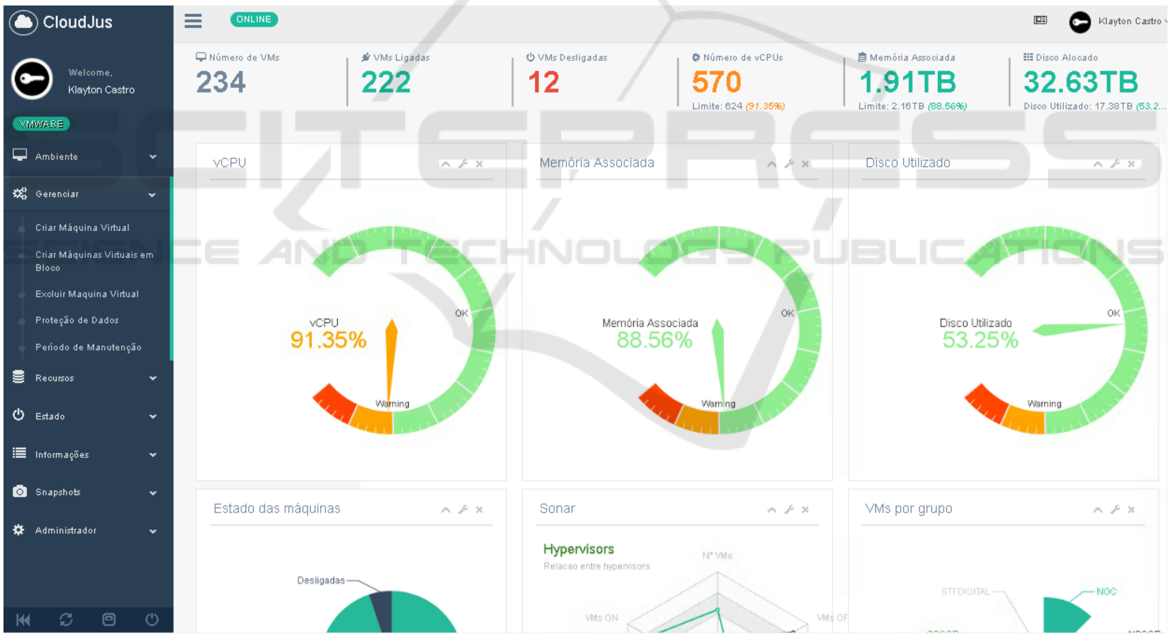

Figure 2: Dashboard of Platform Cloud.Jus.

possible to respond much more quickly to changes than rely on libraries such as libvirt, libcloud, or CMP extensions to integrate with OCCI. During the Cloud.Jus platform deployment at STF, the local virtualization team performed an upgrade on VMware ESXi and vCenter environment, from version 5.5 to 6.5. In just a few minutes, the necessary change for adequacy was implemented easily only in the microservice code associated with this set of hypervisors, not affecting other operations or requiring additional intervention in main system modules. It does not require any changes on middleware, GUI or integration services, that run entirely in a non-intrusive mode, outside the hypervisors. Plugins or extensions are not necessary because we make simple calls to their native APIs. It means that our solution does not require any infrastructure adaptation to support a CMP like CloudStack, OpenNebula or OpenStack, neither change in the local team management way, but exposes a pool of resources of each virtualization node for the community cloud instantly.

The Cloud.Jus architecture focuses on the customer and cloud administrators, unlike other authors who have proposed more provider-centric approaches. The architecture establishes a Management Dashboard that demonstrates the resource utilization and provides rapid interactions with cloud users and cloud administrators, without compromising essential virtualization features at management level, regarding workload movement and allocation of underlying resources, defines block storage among others. This activities, well known by local administrators, does not affect the cloud directly and can be made the same as before. Cloud. Jus infrastructure as a service functional prototype is already able to manage more than 36 virtualization nodes running VMware and Hyper-V hypervisors and hosting more than 300 VMs at 03 geo-distributed datacenters, using OCCI standard and microservices interfaces for the supported hypervisors, VMware, Hyper-V, and OracleVM.

6.2 Control Panel

The Control Panel is a Dashboard that controls the creation, modification, and deletion of VM in the IaaS platform, managing the various components of the architecture through menus and forms. Besides, it presents some useful graphs in top-level monitoring concerning the allocation and utilization of available resources. The programming language chosen was PHP for its simplicity and agility by including a robust set of standardized libraries that make seamless the integration with the LDAP authentication and authorization, as the use of frameworks such as Bootstrap, which brings components visually rich in an implementation based on Gentelella, according to Figure 2. The cloud administrator has maximum management privileges on the Dashboard. It controls the resource quotas for each user groups over the cloud assigned to each hypervisor cluster environment. On the other hand, a cloud user has access limited to its scope, having full control from resource allocation to console execution, which uses the native VMware and Hyper-V implementations. Until now, Cloud.Jus platform does not support Oracle VM console. It is necessary to use SSH protocol to get access to its VMs.

6.3 Orchestrator

The orchestrator came from the need for the IaaS model cloud tools that manage public clouds to perform essential functions of a private cloud, but on a large scale. It provides automated management of the cloud environment control and usage tasks in the VMs (creation, monitoring, updating, and deletion). Depending on the architecture of the tool, it can be seen as a service made available via API or CLI. The orchestration layer automates tasks and procedures in using the environment, coordinating the demands passed through GUI, CLI and API. Back-end modules have been developed to handle integration services between these layers and triggered by the GUI. The implementation was done using Powershell scripts that forward requests to hypervisors through the workers and communicate the actions to middleware, which deals with the update of the metadata repository according to figure 1, which demonstrates the interaction between the components of the orchestration layer and the other infrastructure resources.

6.4 Resources Management

It maintains the operations of available infrastructure resources. It is a responsibility of each member of the community to act on the existing tools in each virtualized solution employed, i.e., VMware vCenter. Middleware keeps updated information about hypervisors allocated resources into metadata NoSQL based repository, which can be queried by other components. For this, we chose Cassandra, because it provides scalability, high performance, and broad applicability as a distributed storage system, supporting dynamic control over data layout and format (Konstantinou et al., 2011). Platform message tasks, asynchronously received via the GUI, API, or CLI interface using the OCCI standard, are necessary. So, that process seamless on back-end nodes (workers), giving to the system a more fluid look. A queue messaging is needed for parallelizing and, especially, to order-dependent tasks, like a shutdown and an increasing memory activity for the same VM. For this, we utilize Kafka as a distributed messaging system for collecting and delivering messages because its low latency, that is suitable for offline and online message consumption in an efficient and scalable way, providing distributed support and scale out design (Kreps et al., 2011). Until now, covers these following IaaS common tasks, among others:

- Creating and Deleting VMs with CentOS 7.6, Windows 2012R2, Oracle EL 7.3 and Ubuntu 16.04 images;
- Creating, Changing and Deleting Resources (persistent disk, memory, CPU);
- VM State operations (initialization, shutdown, and reboot);

- Snapshot Operations;
- Console Support: Provides to users a server administration console to interact with VMware and Hyper-V transparently.
- Integration and Automation: native support for monitoring assets performance, availability and service levels with Zabbix/Grafana built-in;

6.5 Monitoring

It allows the administrator and users to track in general the use of the available and available computing resources. The integration between Cloud.jus and the Zabbix and Grafana tools via API was implemented, which present real-time information about metrics such as CPU usage and queue, memory usage, network traffic, disk consumption, among other parameters.

7 DISCUSSION

One of the factors we want to eliminate with the implementation of interoperability in the proposed community cloud is the vendor lock-in effect. Sometimes open-source alternative could not fully meet the requirements in the scenario presented in the Brazilian Courts, that requires full compatibility with the three main corporate hypervisors (VMware, Hyper-V, and Oracle VM). OpenStack offers limited support for VMware, leaving Oracle VM uncovered. Integration with Hyper-V depends on third-party plugins and does not cover end-to-end management. The integration with VMware to run OpenStack in its latest release requires NSX, that is the manufacturer's SDN module, which is not yet so popular and is not available on most organs. The Horizon GUI interface is visually poor, and it is not simple to implement several modules and components of the solution, requiring deep immersion of the administrators, something that will not always be possible and ending up incurring vendor lock-in, something that you want to avoid.

OpenNebula, while providing an advantageous workflow for the release of features presented by the visually appealing and functional SunStone interface, which, despite its smooth implementation, still lacks in integration. Limited support is offered to VMware, requiring static sockets to establish the console opening through VNC, where a token is the name of the port and the machine. It represents a potential vulnerability when it comes to an environment that needs extend to other bodies and communication via the internet. Hyper-V is not supported, nor is third-party plugins used.

Despite supporting Oracle VM and VMware, CloudStack does not support Hyper-V. Eucalyptus has a different proposal, based on the possibility of porting the environment through the compatibility of its API with Amazon AWS. Neither tool offers integration with robust monitoring tools like Zabbix and legacy support. We described the approaches considered by the main related works in the table 1. There are distinctions about integration and monitoring characteristics can also be observed in these tables, whose functionality can be more oriented to the metrics of the solution itself or flexible metrics to users, in addition to evaluating the coupling of these solutions.

Search for compatibility with the Amazon APIs, and OCCI extensibility is quite recurrent. Approaches such as CYCLONE (Slawik et al., 2017) and FogBow (Brasileiro et al., 2016b) middleware proposals are attractive to enable integration operations, but require the association of other tools to obtain a robust community cloud model. Other alternatives such as Aurora (Wickboldt et al., 2014) and DIRAC FCP (Casaj et al., 2014) do not yet provide a sufficiently decoupled solution to serve more heterogeneous infrastructures, such as the Brazilian PJU.

Considering that an IaaS approach could maintain transparency and compatibility with legacy applications by acting at a lower level, our choice in this project employied an architecture to converge already existing resources into a community cloud with low-coupled design, that can deal with different infrastructure using multiple data centers, network devices, storage systems, physical servers, and hypervisors. It providing instances of VMs that abstract the complexity of the underlying layers in an agile and not intrusive way. Now, it is easier to understand the use of tools as interchangeable components and their role in the process. We developed middleware to interface with different hypervisors, a GUI (Dashboard), a CLI and other integration services, that offers a unified resources view and monitoring automation for the cloud.

8 CONCLUSION

The main contribution of this work is the seamless provisioning and federation middleware, able to integrate components and infrastructure resources with low coupling, keeping the focus on the needs of each Court, besides a unique Dashboard, that centralizes the cloud management and high-level capacity monitoring. So, while avoiding the vendor lock-in effect, it is not required to get loss of features on migrating from traditional to cloud infrastructure. Its client-centric integration and monitoring characteristics use the APIs of each VMware, Hyper-V, and OracleVM enterprise virtualization platform that become more suitable. Unlike other solutions, that depend on evolutions at library level or generic APIs, such as libvirt, libcloud, and OCCI adaptations, making it unfeasible for system upgrades and version updates in large, complex and legacy environments, sacrificing native hypervisors features by making integration to the virtualization layer something very rigid.

Initially, the platform was implemented to integrate the available resources in the two leading institutions of the Brazilian PJU, the Federal Supreme Court (STF) and the National Council of Justice (CNJ). Cloud.Jus also enables a very flexible native event monitoring approach, supporting the common associated systems APIs: Zabbix and Grafana in, allowing both monitoring and proactive and reactive actions. It may facilitate the transition to a cloud computing model in other organizations also, to promote common interests, such as maintenance of security in the geographic distribution of their data, greater redundancy, and tolerance to failures and catastrophes, implementation of mechanisms to guarantee the strategy of continuity of services. We believe Cloud.Jus concepts present a feasible architecture contribution by offering a secure and weakly coupled cloud management solution at the resources provider level. For future work, we intend to implement the console calling feature for Oracle-VM and consider mechanisms to provide billing and resource savings reports.

ACKNOWLEDGEMENT

Klayton Castro thanks the Institutional Security Office of the Presidency of the Republic of Brazil (Grant 002/2017).

REFERENCES

Ahn, J., Park, C., Huh, J., Lewis, J., and Fowler, M. (2014). Microservices. In *Proceedings of the 47th International Symposium on Microarchitecture*.

Baur, D. and Domaschka, J. (2016). Experiences from building a cross-cloud orchestration tool. In *Proceedings of the 3rd Workshop on CrossCloud Infrastructures Platforms*. ACM Press.

Baur, D., Seybold, D., Griesinger, F., Tsitsipas, A., Hauser, C. B., and Domaschka, J. (2015). Cloud orchestration features: Are tools fit for purpose? In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 95–101.

Bégin, M.-E., Jones, B., Casey, J., Laure, E., Grey, F., Loomis, C., and Kubli, R. (2008). An EGEE Comparative Study: Grids and Clouds evolution or revolution?

Brasileiro, F., Silva, G., Araújo, F., Nóbrega, M., Silva, I., and Rocha, G. (2016a). Fogbow: A middleware for the federation of iaas clouds. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 531–534.

Brasileiro, F., Vivas, J. L., Silva, G. F. D., Lezzi, D., Diaz, C., Badia, R. M., Caballer, M., and Blanquer, I. (2016b). Flexible federation of cloud providers: The EUBrazil cloud connect approach. In *Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016*.

Buyya, R. (2009). Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. *Proceedings: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*, page 1.

Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition. 9780124095397, 9780124114548.

Casaj, A., Diaz, R. G., Tsaregorodtsev, A., Méndez Muñoz, V., Casajús Ramo, A., Diaz, R. G., and Tsaregorodtsev, A. (2014). Cloud governance by a credit model with DIRAC. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, pages 679–686. SciTePress - Science and and Technology Publications.

Ciuffoletti, A. (2014). A simple and generic interface for a cloud monitoring service. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*. SciTePress - Science and and Technology Publications.

Cunha Rodrigues, G., Calheiros, R. N., Guimarães, V. T., Santos, G. L., Carvalho, M. B., Granville, L. Z., Tarouco, L. M. R., and Buyya, R. (2016). Monitoring of cloud computing environments. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*.

Demchenko, Y., Makkes, M. X., Strijkers, R., and De Laat, C. (2012). Intercloud Architecture for interoperability and integration. In *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*.

Dukarić, R. and Jurič, M. B. (2013). A Taxonomy and Survey of Infrastructure-as-a-Service Systems. *Lecture Notes on Information Theory*.

Edwards, D. (2012). The History Of DevOps. http://itrevolution.com/the-history-of-devops.

Edwards, D. (2014). Introducing DevOps to the Traditional Enterprise. *InfoQueue / eMag Issue*, 14.

Fiore, S., Mancini, M., Elia, D., Nassisi, P., Brasileiro, F. V., and Blanquer, I. (2015). Big data analytics for climate change and biodiversity in the EUBrazilCC federated cloud infrastructure. In *Proceedings of the 12th*

ACM International Conference on Computing Frontiers - CF '15.

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

Jamsa, D. K. (2012). *Cloud Computing: SaaS, PaaS, IaaS, Virtualization, Business Models, Mobile, Security and More*. Jones & Bartlett Learning. Http://web.mit.edu/smadnick/www/wp/2013-01.pdf – .

Kim, G., Willis, J., Humble, J., and Debois, P. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability and Security in Technology Organizations*, volume 2. T Revolution, 2nd edition edition.

Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., and Koziris, N. (2011). On the elasticity of nosql databases over cloud management platforms. In *Proceedings of the 20perscriptth ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2385–2388, New York, NY, USA. ACM.

Korte, F., Challita, S., Zalila, F., Merle, P., and Grabowski, J. (2018). Model-driven configuration management of cloud applications with OCCI. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science*. SCITEPRESS - Science and Technology Publications.

Kreps, J., Narkhede, N., and Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of 6th International Workshop on Networking Meets Databases (NetDB)*. IEEE.

Llorente, I. M. (2013). Eucalyptus, CloudStack, OpenStack and OpenNebula: A Tale of Two Cloud Models.

Lynn, T., Hunt, G., Corcoran, D., Morrison, J., and Healy, P. (2015). A comparative study of current open-source infrastructure as a service frameworks. In *Proceedings of the 5th International Conference on Cloud Computing and Services Science*. SciTePress - Science and and Technology Publications.

Microsoft (2016). Hyper-V Technology Overview. https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview.

OCCI (2018). Open Cloud Computing Interface. http://occi-wg.org/.

Oracle (2018). Oracle VM Overview. https://www.oracle.com/technetwork/server-storage/vm/overview/index.html.

Par, B., Kimle, M., Fern, P. O., Parák, B., Šustr, Z., Kimle, M., Fernández, P. O., García, Á. L., Sachtouris, S., and Muñoz, V. M. (2016). Evolution of the Open Cloud Computing Interface. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, volume 2, pages 339–346.

Parmar, H. and Champaneria, T. (2014). Comparative Study of Open Nebula, Eucalyptus, Open Stack and Cloud Stack. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(2):991–996.

Parpaillon, J., Merle, P., Barais, O., Dutoo, M., and Paraiso, F. (2015). OCCIware - A formal and tooled framework for managing everything as a service. In *CEUR Workshop Proceedings*.

Paul, S., Jain, R., Samaka, M., and Pan, J. (2014). Application delivery in multi-cloud environments using software defined networking. *Computer Networks*, 68:166–186.

Petcu, D., Cr, C., and Rak, M. (2013). On the Interoperability in Multiple Clouds. In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science*. SciTePress - Science and and Technology Publications.

Roveda, D., Vogel, A., and Griebler, D. (2016). Understanding, Discussing and Analyzing the OpenNebula and the OpenStack IaaS Management Layers.

Sefraoui, O., Aissaoui, M., and Eleuldj, M. (2012). Openstack: Toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42. Full text available.

Sessions, R. (2006). Um melhor caminho para arquiteturas empresariais.

Slawik, M., Blanchet, C., Demchenko, Y., Turkmen, F., Ilyushkin, A., Laat, C. D., and Loomis, C. (2017). CY-CLONE: The multi-cloud middleware stack for application deployment and management. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, pages 347–352.

Sosinsky, B. (2010). *Cloud Computing Bible*, volume 762. John Wiley & Sons.

Sotomayor, B., Montero, R. S., Llorente, I. M., and Foster, I. (2009). Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 13:5.

Šustr, Z., Scardaci, D., Sitera, J., Parák, B., and Méndez Muñoz, V. (2016). Easing Scientific Computing and Federated Management in the Cloud with OCCI. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, volume 2, pages 347–354. SciTePress.

Toosi, A. N., Calheiros, R. N., and Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, 47(1):1–47.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A Break in the Clouds: Towards a Cloud Definition. *Computer Communication Review*, 39(1):50–55.

VMware (2018). vSphere ESXi Hypervisor. https://www.vmware.com/br/products/esxi-and-esx.html.

Wickboldt, J. A., Esteves, R. P., de Carvalho, M. B., and Granville, L. Z. (2014). Resource management in iaas cloud platforms made flexible through programmability. *Computer Networks*, 68:54–70. Communications and Networking in the Cloud.