Cloud of Things

Marcus Vinícius Paulino Gomes

Instituto Superior Técnico
marcus.paulino.gomes@tecnico.ulisboa.pt

Table of Contents

Abstract. Abstract should have 200 words at maximum.

1 Introduction

This section will describe the context of this theme, namely:

- What is the scope of the theme?
- What is the actual state of the theme?
- What are the main difficulties confronted?

1.1 Internet of Things

The Internet of Things (IoT) is a concept in which the virtual world of information technology integrates seamlessly with the real world of things [1]. Through the amount of computer and network devices available nowadays, the real world becomes accessible to business and everyday scenarios. A more precise definition of what is Internet of Things was formulated in the Strategic Research Agenda of the Cluster of European Research Projects on the Internet of Things (CERP-IoT 2009):

"*Internet of Things (IoT) is (...) a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network. In the IoT, 'things' are expected to become active participants in business, information and social processes where they are enabled (...) by exchanging data and information sensed about the environment, while reacting autonomously to the 'real/physical world' events and influencing it by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these 'smart things' over the Internet (...) taking into account security and privacy issues.*"

IoT provides access to a more detailed information, which results that the level of analysis can be performed in a large-scale perspective, as well in a small-scale perspective. But Internet of Things is more than a tool for managing business processes more efficiently and more effectively, IoT will also enable a more convenient life for all peoples. Recently, IoT became relevant to industry and end-users [1], mainly because the reduction of cost and miniaturization of technologies used in IoT applications, such as RFID, sensor networks, NFC and wireless communication. Technologies like these allows the detection of status of things, which together with collection and processing of detailed data, allows to create an interactive and responsive network with huge potential for citizens, consumers, business and where is possible to gather immediate responses to events that occurs in the real world [1]. One of the expectations regarding of IoT concerns with real-world awareness provided to information systems [2]. An example is in logistics applications, by the use of RFID, companies can react promptly to relevant physical events, and then manage their processes in a better

way, typically increasing efficiency and reducing costs. Another expectation of IoT concerns with providing services to the end-users through common objects, in that way products will be able to provide recommendations for use and maintenance instructions, supply warranty information or highlight complementary products.

1.2 Cloud Computing

Cloud Computing is a paradigm where its infrastructure is designated as a *"Cloud"* from which its resources are available to businesses and users from anywere in the world on demand [3]. This paradigm enables rapid service delivery in a dinamically scalable and virtualized manner. Actually, clouds are builted on top of modern data centers, that incorporated Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [4], as illustrated on Figure 1. Cloud providers, such as Amazon Web Services[1] , Google Cloud Platform[2] and Microsoft Azure[3] offers these services as utilities in a pay-per-use model in which *QoS* guarantees are offered by means of customized Service Level Agreements (SLA's) [5], that are negotiated between the customers and the Cloud providers.

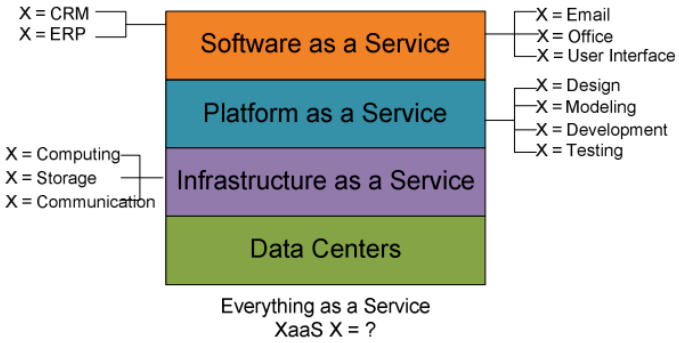

Fig. 1. Hierarchical View of Cloud Computing.

Data Centers allocates all the hardware that supports the cloud, they consist of thousand of physical machines that provides redundancy and ensure reliability in case of site failures. By using the IaaS pattern, Cloud Computing is able to virtualize resources as computing power, storage and network connectivity of the data centers. These computational resources are provisoned on demand in

[1] aws.amazon.com
[2] cloud.google.com
[3] azure.microsoft.com/

a form of Virtual Machines (VM's) - *definition* - deployed in a cloud provider data-center [6]. Cloud also assists application design, development, application hosting and testing by providing a development platform in order to assist the execution of these tasks, this development as a service pattern is know as PaaS. Another pattern supported by Cloud Computing is SaaS. This pattern allows that a single piece of software is transferred to millions of user through a browser [7]. This pattern is beneficial by both users and developers, the developers only needs to maintain a single program while the users can save costs in servers and software.

In a global analysis Cloud computing offers several benefits such as high-availability, high-scalability, flexibility, on-demand service provisoning and massive reduction of costs. By converging IoT applications and Cloud computing, it is possible to take advantage of all these benefits.

1.3 Smart Place

In the context of this work, a smart place can be defined as an ecosystem composed by smart objects such as RFID tags and sensors, that are able to acquire knowledge about this environment and also to adapt this inhabitants in order to improve their experience in that environment [8]. In particular, the deployment of IoT applications in a smart place is a challenge today due the heterogeneity of the smart objects that are present in a smart place and also because the required infrastructure. In order to decrease the complexity of the deployment of an IoT application in a smart place, the Cloud computing paradigm allows to virtualize the required physical infrastructure. Nowadays the infrastructure needed by IoT applications can be virtualized by Cloud providers, which helps to gain more flexibility and reduce the costs in the deployment of an IoT application. Therefore, another challenge concerns with the heterogeneity of a smart place, more precisely the variety of smart objects that can be inside of a given place. Many of these objects uses different communication protocols and drivers to communicate with devices, thus the configuration of this objects must be manually handled, which makes the integration of these objects in the deployment of IoT applications in a smart place a inefficient process.

1.4 Cloud Orchestration

Due the heterogeneity of IoT applications infrastructure service management tasks such as deployment, driver installation and gateway configuration are still handled manually in a particular way for each case. In order to reduce the complexity of this tasks, Cloud Orchestration tools can be used to automate these management tasks. The process of orchestration woven the software components of the application in a one piece that can be managed more effectively, in order to ensure a smooth and fast service delivery. Cloud Orchestration helps to take advantage of the full benefits of Cloud computing by providing features that can include:

– Simplify, automate and optimize service deployment by integrating cloud capabilities across heterogeneous environments.
– Automated high-scale provisioning and de-provisioning of resources with policy-based tools.
– Real-time monitoring of physical and virtual Cloud resources.
– Selection of Cloud services such as storage and hosting, through a self-service portal.
– Real-time monitoring of usage and accounting chargeback capabilities to track and optimize system usage.

In addition, orchestration allows to reduce significantly the costs related with labor and resources, since that manual intervention and management of varied IT services and resources are not needed. Actually several tools to performing Cloud Orchestration are available in the market such as IBM Cloud Orchestrator[4], HP Operations Orchestration[5] and GigasSpaces Cloudify[6] and some open-source tools like Ubuntu Juju[7] and OpenTOSCA[8].

1.5 Fosstrak

Fosstrak is an open-source RFID project[9] that implements the EPC (Electronic Product Code) Network specifications. Fosstrak is composed by modules that allows to store and query about events captured by a RFID reader, convert the EPC identifiers to others EPC representations, filter and collect the data from RFID readers and to configure and manage RFID readers. These modules provides an entire RFID infrastructure that accelerates the development of IoT applications and allows to users to gain hands-on experience with the EPC Network.

The rest of the paper is organized in the following way ...

[4] ibm.com/software/products/en/ibm-cloud-orchestrator
[5] hp.com/us/en/software-solutions/operations-orchestration-it-process-automation
[6] gigaspaces.com/cloudify-cloud-orchestration/overview
[7] juju.ubuntu.com
[8] iaas.uni-stuttgart.de/OpenTOSCA/indexE.php
[9] https://code.google.com/p/fosstrak/

2 Objectives

The life-cycle of a Cloud-based IoT application is composed by several states, as illustrated in the Figure 2.

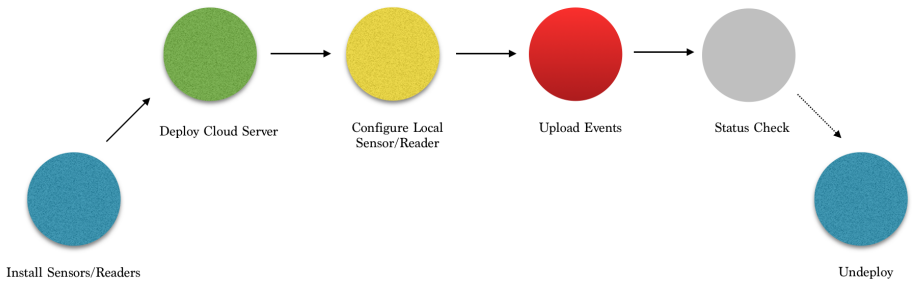

Fig. 2. IoT application life-cycle.

Thus, the objective of this work is to decrease the complexity of deployment and management of Cloud-based IoT applications in a smart place. Usually the deployment of such applications is performed by a technician that must manually configure and install the components of the application. In order to reduce the complexity of this process, the most effective approach is to automate the deployment process of the application. That is achieved by performing the deployment through Cloud Orchestration tools, that allows to specify the components and the relations between themselves in a high-level perspective and also provisioning the necessary resources at the Cloud in a effective way. These tools also allows to perform the monitoring of these applications during its life-cycle as well undeploying them. However, this tools don't solve all our problems. In particular, IoT applications require a software stack that usually is composed by a database, a web server and the event processing software, in our case the Fosstrak project. But orhcesratation tools normally lacks the integration of the event processing software with the other components. Thus, to support the integration of such componets these tools must be extend to support the event processing software.

However, managing these applications through a Orchestrator tool requires an elevated technical knowledge. To allow that non-technical users can perform the managing operations having in perspective the high-level business rules of the smart place, the main objective of this work is to permit that non-technical users can perform the management of smart spaces only having in mind the business rules of that particular place. To achieve that, these high-level rules must be translated to a more low-level rules that can be expressed in terms of non-functional requirements that can be used to estimate the resources needed by the application in order to have an acceptable *QoS*. In the other hand, to allow those non-technical users to monitoring the smart space, these low-level

rules must be translated to a high-level rules that can be expressed in terms of business rules of a given smart space. For instance, if a smart place has a flow of people of 500 persons per day, that business rule will be translated to non-functional requirements that allows to calculate the amount of storage and bandwidth required to ensure the QoS of the application. Furthermore, by monitoring the service level offered by the Cloud providers it will be possible to determine if the Cloud is overloaded with the amount of data generated by the smart space or not.

3 Related Work

Cloud and Internet of Things are emerging computing paradigms that features distinctively different computing resources and system architecture. As its popularity has been growing across the academics and the industry, researchers and developers are spending a lot of effort to investigate how to integrate these technologies in order to take advantage of the benefits provided by both of them.
IoT applications often encapsulate several relatively complex protocols and involves different software components. Moreover, they require a significant investment in infrastructure, besides that the system administrators and users spend time with large client and server installations, setups, or software updates. As most of the computing resources are allocated on the Internet on servers in Cloud computing, integrating these paradigms in a Cloud-based model results in a solution with more flexibility of implementation, high scalability and high availability, and with a reduced upfront investment.

3.1 Cloud Computing and Internet of Things

In RFID-based IoT applications, Guinard at el. [9] point out that the deployment of RFID applications are cost-intensive mostly because they involve the deployment of often rather large and heterogeneous distributed systems. As a consequence, these systems are often only suitable for big corporations and large implementations and do not fit the limited resources of small to mid-size businesses and small scale applications both in terms of required skill-set and costs. To address this problem, Guinard at el. proposes a Cloud-based solution that integrates virtualization technologies and the architecture of the Web and its services. The case of study consists in a IoT application that uses RFID technology to substitute existing Electronic Article Surveillance (EAS) technology, such as those used in clothing stores to track the products. In this scenario they applied the Utility Computing blueprint to the software stack required by the application using the AWS platform and the EC2 service. The Elastic Cloud Computing (EC2) service allows the creation and management of virtual machines (Amazon Machine Images, or AMIs) that can then be deployed on demand onto a pool of machines hosted, managed and configured by Amazon. A benefit of this approach is that the server-side hardware maintenance is delegated to the cloud provider which is often more cost-efficient for smaller businesses. Furthermore, it also offers better scaling capabilities as the company using the EPC Cloud AMI, can deploy additional and more powerful instances regarding to the amount of requests.

Distefano [10] at el. proposed a high-level modular architecture to implement the Cloud of Things. According to Distefano at el. things not only can be discovered and aggregated, but also provided as a service, dynamically, applying the Cloud provisioning model to satisfy the agreed user requirements and therefore establishing Things as a Service providers. The *Things as a Service* (TaaS)

paradigm envisages new scenarios and innovative, pervasive, value-added applications, disclosing the Cloud of Things world to customers and providers as well, thus enabling an open marketplace of "things". To address this issues, an ad-hoc infrastructure is required to deal with the management of sensing an actuation, mashed up resources provided by heterogeneous Clouds, and things, by exploiting well know ontologies and semantic approaches shared by and adopted by users, customers and providers to detect, identify, map and transform mashed up resources. The proposed architecture provides blocks to deal with all the related issues, while aiming to provide things according to a service oriented paradigm.

CloudThings [11] is an architecture that uses a common approach to integrate Internet of Things and Cloud Computing. The proposed architecture is an online platform which accommodates IaaS, Paas, and SaaS and allows system integrators and solution providers to leverage a complete IoT application infrastructure for developing, operating and composing IoT applications and services. The applications consists of three major modules, the CloudThings service platform, that is a set of Cloud services (IaaS), allowing users to run any applications on Cloud hardware. This service platform dramatically simplifies the application development, eliminates need for infrastructure development, shortens time to market, and reduces management and maintenance costs. The CloudThings Developer Suite is a set of Cloud service tools (PaaS) for application development, such as Web service API's, which provide complete development and deployment capabilities to developers. The CloudThings Operating Portal is a set of Cloud services (SaaS) that support deployment and handle or support specialized processing services. To evaluate CloudThings, a smart home application based on a Cloud infrastructure was implemented. In the application, the sensors read the home temperature and luminosity and the Cloud application stores and visualized them, so that the user can view the smart home temperature and luminosity anywhere. In particular, in these implementation the Cloud architecture was extended by inserting a special layer for dynamic service composition. This middleware encapsulates sets of fundamental services for executing the users service requests and performing service composition, such as process planning, service discovery, process generation, process execution, and monitoring. This first implementation also demonstrates that this middleware as a service releases the burden of costs and risks for users and providers in using and managing those components.

3.2 Automated Deployment of IoT Applications

The effort put in the research to integrate the paradigms of Cloud Computing and Internet of Things resulted in a essential contribution, but there are several issues regarding to the integration between Cloud Computing and Internet of Things that must be addressed. In particular, due of the heterogeneity of the

IoT applications environments, its hard for solution providers to efficiently deploy and configure applications for a large number of users. Thus, automation for the management tasks required by IoT applications is a key issue to be explored.

TOSCA (Topology and Orchestration Specification for Cloud Applications) [12] is proposed in order to improve the reusability of service management processes and automate IoT application deployment in heterogeneous environments. TOSCA is a new cloud standard to formally describe the internal topology of application components and the deployment process of IoT applications. The structure and management of IT services is specified by a meta-model, which consists of a *Topology Template*, that is responsible to describe the structure of a service, then there are the *Artifacts*, that describes the files, scripts and software components necessary to be deployed in order to run the application, and finally the *Plans*, that defined the management process of creating, deploying and terminating a service. The correct topology and management procedure can be inferred by a TOSCA environment just by interpreting te topology template, this is known as "declarative" approach. Plans realize an "imperative" approach that explicitly specifies how each management process should be done. The topology templates, plans and artifacts of an application are packaged in a Cloud Service Archive (.csar file) and deployed in a TOSCA environment, which is able to interpret the models and perform specified management operation. These .csar files are portable across different cloud providers, which is a great benefit in terms of deployment flexibility. To demonstrate its feasibility TOSCA was used to specify a typical IoT application in building automation, an Air Handling Unit (AHU). The common IoT components, such as gateways and drivers will be modeled, and the gateway-specific artifacts that are necessary for application deployment will also be specified. By archiving the previous specifications and corresponding artifacts into a .csar file, and deploying it in a TOSCA environment, the deployment of AHU application onto various gateways can be automated. As a newly established standard to counter growing complexity and isolation in cloud applications environments, TOSCA is gaining momentum in industrial adoption as well academic interests.

Breitenbücher at el. [13] proposed to combine the two flavors of management supported by TOSCA, *declarative processing* and *imperative processing*, in order to create a standards-based approach to generate provisioning plans based on TOSCA topology models. The combination of both flavors would enable applications developers to benefit from automatically provisioning logic based on declarative processing and individual customization opportunities provided by adapting imperative plans. These provisioning plans are workflows that can be executed fully automatically and may be customized by application developers after generation. The approach enables to benefit from strengths of both flavors that leads to economical advantages when developing applications with TOSCA. The motivating scenario that is used to evaluate this approach consists in a LAMP-based TOSCA application to be provisioned. The application imple-

ments a Web-shop in PHP that uses a MySQL database to store product and customer data. The application consists of two application stacks, one provides the infrastructure for the application logic and the other hosts the database, were both will be run on Amazon's public IaaS offering Amazon EC2. To measure the performance of the deployment using the two-flavor approach, the strategy adopted was measure the time spent to generate provisioning plans regarding the number of templates required by the application. The results indicates that the required time increases linearly to the number of templates.

Recently a growing number of organizations are developing Orchestrators, Design Tools and Cloud Managers based on TOSCA. Juju is an Open Source TOSCA Orchestrator that can deploy workloads across public, private clouds, and directly onto bare metal. HP Cloud Service Automation is cloud management solution that supports declaratives services design that are aligned with TOSCA modeling principles. GigaSpaces Cloudify orchestrates TOSCA Service Templates using workflows to automate deployments and other DevOps automation processes. IBM Cloud Orchestrator provides integrated tooling to create TOSCA applications, deploy them with custom polices, monitoring and scale them in cloud deployments.

3.3 Service Level Agreements in IoT Applications

In order to guarantee an acceptable *QoS* for a Cloud applications, Service-Level Agreements (SLA's) must be defined by the customers in order to monitor the performance of the applications. Regarding Cloud-Based IoT applications, SLA's can be defined with two purposes, to ensure that the events are been correctly captured and to ensure that the available resources are sufficient for the application runs with the desired performance.

Waizenegger at el. [14] proposed a mechanism to demonstrate how non-functional requirements are defined in TOSCA using policies. TOSCA allows to specify non-functional requriements like cost, security, and environmental issues. However, TOSCA lacks detailed description of how to apply, design, and implement policies. This paper proposes two mechanisms for processing policies during the deployment and management of Cloud services in TOSCA. The policies are described according a taxonomy were a Cloud service policy is a tuple of elements that describe its behavior and effect. To perform the implementation of this policy-specific logic two approches was proposed, an Implementation Artifact Based Policy Enforcement (IA-Approach) and a Plan-Based Policy Enforcement (P-Approach). In the IA-Approach the existing Implementation Artifacts - responsible to perform service management operations - are extended to support policy enforcement implementations, thus these artifacts are comprised of two alternative implementations for each service management operation: one that is policy enforcing and one that is not. In the P-Approach the policies are enforced in an imperative way, for each policy the plan execute the appropriate steps to

enforcing the policy. Altough both of these approaches allows the enforcement of policies, the approach used for realizing this should be chosen with caution. The IA-Approach is more suitable for less complex scenarios and the policies implemented with this apporach can be reused very easily. On the other hand, the enforcement of policies in more complex scenarios should be performed according the P-Approach, that allows to have a global control of the triggered enforcement actions.

Breitenbücher et al. [15] proposed an approach that enables to automate the provisioning and management of composite Cloud applications in compliance with non-functional security requirements defined by policies. In this approach the Management Planlet Framework [16] was extended to support policy-aware provisioning and management based in Management Policies that binds non-functional security requirements to the management tasks that must enforce them. This paper also introduces the concept of Management Annotation Policies, which defines the semantics and how the policy must be processed. The automation of provisioning and management tasks are performed by Policy-aware Management Planlets, that executes this tasks considering the Annotation Policies attached to each operation. The Framework also allows the Cloud providers and application developers to specify their own policy-aware management logic in a flexible and reusable manner independently from individual applications. The management logic can be defined in two ways, by applying Automated Management Patterns or by manually creating Desired Application State Models. The Framework was evaluated in terms of feasibility, performance, economics, limitations and extensibility. To validate the concept technically, a prototype was sucessfully implemented based on a previous implementantion of the Management Planlet Framework.

4 Solution Architecture

Actually a typical scenario for an IoT application consists of a smart space that contains several smart objects and all the infrastructure required to support this application, namelly RIFD tags, sensors, readers and servers, as illustrated at the Figure 3. This scenario presents several issues regarding the deployment of the application, the low scalability, the costs of infrastructure and the maintenance of the same.

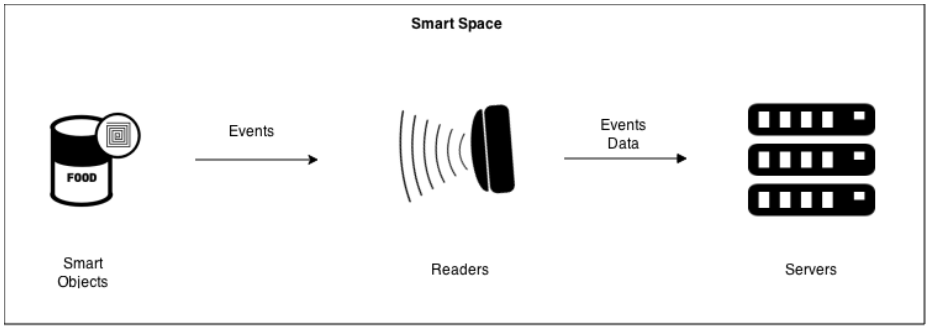

Fig. 3. Typical smart space scenario.

By converging the IoT applications with the Cloud Computing paradigm the objective is simplify this scenario by leveraging the required infrastructure by these applications to the Cloud providers, as illustrated at the Figure 4. Furthermore, the convergence of this two paradigms, allows to take advantage of the benefits offered by Cloud computing as referenced in **Section 1.2**.

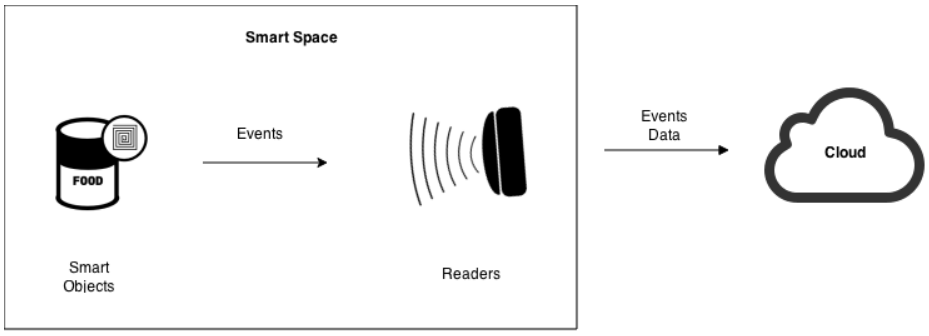

Fig. 4. Cloud-based smart space.

4.1 State of Art

Actually Cloud-based IoT applications are the state of art of this kind of solutions. By leveraging the infrastructure to the Cloud providers, these applications has a high-availabity, can dinamically scale while spending a fraction compared with the tradional solutions. However, as earlier mentioned the deployment of such applications still is an issue, due its complexity and required manual intervention. The deployment of Cloud-based IoT applications usually are performed through IT automation tools, such as Chef[10] and Puppet[11]. These tools enables to automate the deployment of such applications in a certain way, given that all the components of the application and the relation between themselves must be specified manually, which requires considerable manual work and expertise by the person that is performing the deployment. However, the deployment process of these solutions is not the only existent issue. Monitoring the application life-cycle is a task that requires a lot of effort and expertise by the system admnistrators.

In order to solve this problem, the adopted approach relies on perform the deployment of these applications by using Cloud Orchestrator tools. As mentioned in **Section 1.4**, these tools allows to specify the application components and their relations in a high-level perspective and to execute the management tasks required by the application during its life-cycle. As a matter of fact, in a low-level perspective cloud orchestration tools express the high-level perspective defined by the user into scripts that latter are executed using IT automation tools such as Puppet and Chef.

4.2 Cloud of Things Architecture

The main objective of Cloud of Things decrease the complexity of deployment and management of IoT applications. To achieve this objective, Cloud of Things must enable non-technical users - the business managers - to perform the monitoring of Cloud-based IoT applications as well defining Service Level Agreements in a high-level way. In order execute this tasks, users must be able to interact with the application that is running at the cloud in order to observe their state and to apply some decisions based on the performance of the application. Thus, Cloud of Things must provide a service that allows the users to perform such actions. At Figure 5 we present the Cloud of Things architecture.

[10] www.chef.io
[11] puppetlabs.com

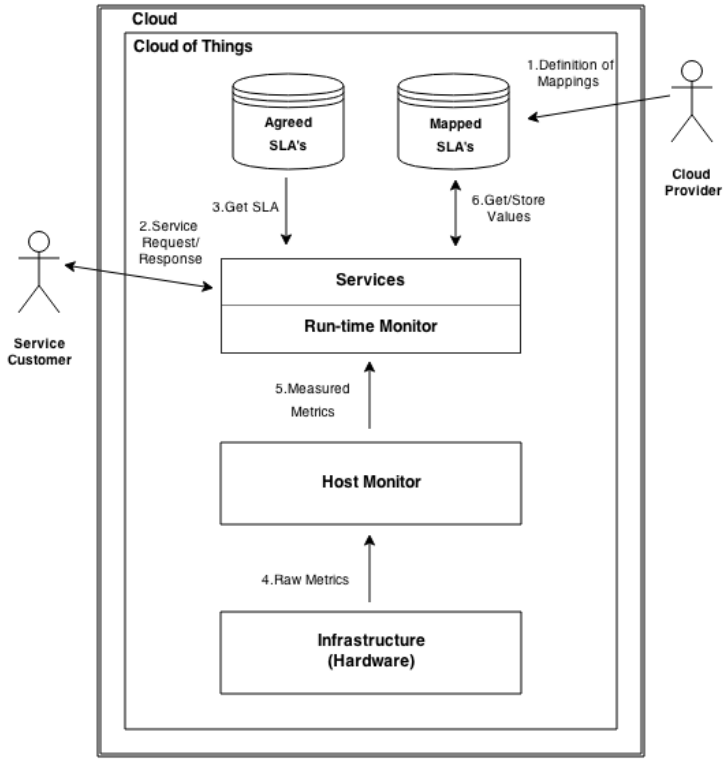

Fig. 5. Cloud of Things Architecture.

The presented architecture is based on the LoM2HiS Framework [17] architecture. In this architecture the *Services* component and the *Run-time Monitor* represents the application layer where services are deployed using a Web Service container. The *Run-time Monitor* is responsible to monitor the services based on the negotiated and agreed SLAs. After the Cloud provider agrees on the SLA terms, the agreed SLAs are stored in the repository for service provisioning and the following steps are executed:

1. The Cloud provider creates rules for the framework mappings using Domain Specific Languages[12] (DSL's).
2. The customer requests the provisioning of an agreed service.
3. Once the request is received, the run-time monitor loads the service SLA from the agreed SLA repository.
4. The resource metrics are measured by monitoring agents, these metrics are stored in a raw format that later are accessed by the host monitor.

[12] Domain Specific Languages are small languages that normally are tailored to a specific problem domain.

5. The host monitor extracts metric-value pairs from the raw metrics and them transmits them periodically to the run-time monitor.
6. After receive the low-level metrics, the run-time monitor uses predefined mapping rules to map the low-level metrics into a equivalent form of the agreed SLA and them the resulting map is stored in the mapped metrics repository.

In this architecture, the *Run-time monitor* uses the mapped values to monitor the status of the deployed services. Once it detects that a SLA is violated, the *Run-time monitor* must alert the customer of the violated SLA. At this point the customer is responsible to take the decisions in order to correct the state of the system.

5 Evaluation Methodology

To evaluate the solution the approach consists in evaluate the performance of the Cloud and also the correctness of the received RFID events.

RFID has its particular characteristics, which makes that traditional event processing systems cannot support them. Furthermore, RFID events are temporal constrained [18]. Temporal constraints as the time interval between two events and the time interval for a single event are critical to event detection. In order to assure the correctness of these received events, some conditions must be defined regarding the time interval that between the events.

Another aspect that is important concerns with the performance of Cloud regarding the amount of data generated by the events. Cloud computing creates an illusion that available computing resources on demand are infinite [19]. However, with the increase of the amount of events, we must measuring these computing resoruces in order to determine if the system fulfilling the QoS requirements. To achieve that, some metrics such as CPU utilization, I/O throughput, memory RAM consumption and memory storage must be measured and compared with the defined SLA's in order to decide if the system is overloaded or not.

6 Conclusion

This section will present the general conclusion of the work realized.

A Planning

References

1. D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer Publishing Company, Incorporated, 1st ed., 2011.
2. F. Mattern and C. Floerkemeier, "From the internet of computers to the internet of things," in *From active data management to event-based systems and more*, pp. 242–259, Springer, 2010.
3. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.
4. W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pp. 684–689, IEEE, 2010.
5. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.
6. B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.
7. S. Zhang, S. Zhang, X. Chen, and X. Huo, "Cloud computing research and development trend," in *Future Networks, 2010. ICFN'10. Second International Conference on*, pp. 93–97, IEEE, 2010.
8. D. Cook and S. Das, *Smart environments: technology, protocols and applications*, vol. 43. John Wiley & Sons, 2004.
9. D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, rest and mashups to simplify rfid application development and deployment," in *Proceedings of the Second International Workshop on Web of Things*, p. 9, ACM, 2011.
10. S. Distefano, G. Merlino, and A. Puliafito, "Enabling the cloud of things," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 858–863, IEEE, 2012.
11. J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, and H. Jin, "Cloudthings: A common architecture for integrating the internet of things with cloud computing," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pp. 651–657, IEEE, 2013.
12. F. Li, M. Vogler, M. Claeßens, and S. Dustdar, "Towards automated iot application deployment by a cloud-based approach," in *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on*, pp. 61–68, IEEE, 2013.
13. U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, and J. Wettinger, "Combining declarative and imperative cloud application provisioning based on tosca," *IC2E. IEEE*, 2014.
14. T. Waizenegger, M. Wieland, T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, B. Mitschang, A. Nowak, and S. Wagner, "Policy4tosca: A policy-aware cloud service provisioning approach to enable secure cloud computing," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pp. 360–376, Springer, 2013.
15. U. Breitenbücher, T. Binz, C. Fehling, O. Kopp, F. Leymann, and M. Wieland, "Policy-aware provisioning and management of cloud applications," *International Journal On Advances in Security*, vol. 7, no. 1 and 2, pp. 15–36, 2014.

16. U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and M. Wieland, "Policy-aware provisioning of cloud applications," in *SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies*, pp. 86–95, 2013.
17. V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level slas-lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pp. 48–54, IEEE, 2010.
18. F. Wang, S. Liu, P. Liu, and Y. Bai, "Bridging physical and virtual worlds: complex event processing for rfid data streams," in *Advances in Database Technology-EDBT 2006*, pp. 588–607, Springer, 2006.
19. M. Armbrust, O. Fox, R. Griffith, A. D. Joseph, Y. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "M.: Above the clouds: A berkeley view of cloud computing," 2009.