Intelligent Agent-Based Approach for Real-Time Reconfiguration of Cloud Application

Walid Bouzayen*†, Hamza Gharsellaoui*‡, Mohamed Khalgui*§
*LISI, INSAT Institute, Carthage University, Tunisia
†FST, Tunis El Manar University, Tunisia
‡ENI-Carthage, Carthage University, Tunisia
§SystemsControl Lab, Xidian University, China
Email: {walid.bouzayen, gharsellaoui.hamza, mohamed.khalgui}@gmail.com

***Abstract*—This paper deals with the problem of reconfiguration of Internet of Things (IoT) and Cloud Computing (CC) applications which have been widely studied recently. They are composed of a set of interconnected software components running on real-time on remote virtual machines. Virtualization is one of the building blocks for cloud computing and provides the mechanisms to implement the dynamic allocation of resources. Once cloud applications are deployed, they need to be reconfigured in order to react to any disturbance created by the removal or modification of virtual machines or components that make up these machines. In this paper, the original approach proposed for handling these reconfiguration scenarios must preserve the application consistency, feasibility, computing time, low power consumption and respect important architectural invariant related to real-time properties and to software dependencies. Finally, the challenges, advantages of the proposed cloud architecture and future works for the application and implementation are discussed.**

Keywords–Cloud Computing; Reconfiguration Technology; Internet of Things.

I. INTRODUCTION

This work touches areas of Internet of Things (IoT) and Cloud Computing (CC). A system is made of interconnected Virtual Machines (VMs) where each one runs a set of processes. It is usually necessary to reconfigure a process, component or any object of the cloud, which also requires the reconfiguration of the whole system at run-time. CC aims to build a virtual infrastructure providing users with remote computing and storage capacity. CC is rapidly gaining traction in industrial and business ethics. It offers businesses online services on demand and allows them to reduce costs on software, hardware and information technology support [2]. On one hand, dealing with this new technology, CC analyzes the informational duties of hosting companies that own and operate CC datacentres (e.g., Amazon). On the other hand, it considers the cloud services providers leasing ”space in the cloud” from hosting companies (e.g., Salesforce, Dropbox) and it examines the private ”clouders” and the business using these services.

IoT is a concept of communication between people and smart objects and the CC technology is based on the virtualization. In literature, Srivastava [15] states that IoT represents the greatest level of technological convergence that we can currently imagine. Moving from an Internet of user-generated content to thing-generated content will produce a new level of sensory awareness, with the ultimate goal of increasing our control over time and space. The vision that characterizes the IoT is one where things are connected that we would not previously have considered relevant to computation or feasible to integrate with a network. This change will allow various forms of data to be collected from all kinds of objects around us, from the use of appliances and lighting to door locks, clothes or toothbrushes [16]. Therefore, in the domain of CC, we must deal with two key issues: computing and storage. This latter is increasingly considered as a major research field in the area of CC. Indeed, more and more data intensive applications are attracted by the cloud since cloud storage can provide high scalability, fault tolerance, security, availability and cost-effective data services [1]. The huge number of users of the cloud and the devices makes the control of the performance of different storage nodes very difficult or impossible to the network manager. The cloud network has a huge amount of data written to it by different users on storage devices; a great percentage of this data might be similar or identical. Almost 90% of the data stored in cloud is duplicated [3].

In order to optimize the storage over cloud, many methods and techniques have been used: compression, snapshots and deduplication. Data compression [24] is a technique to reduce storage cost by eliminating redundancies in different files. There exist two types of compression, lossy and lossless. The technology of Snapshot [21] is defined as a virtual copy of a set of files. This technique solves several data backup problems, including backing up large amounts of data and recovering corrupted data. Data deduplication [22] is a technique for reducing the amount of storage space by eliminating redundant data. This is called intelligent compression. Many works related to the cloud reconfiguration have been proposed. Duran and Salaun [20] proposed a reconfiguration scenario on a simple Web application that includes three VMs with the following components: Apache, Tomcat, MySQL. The first drawback is that the reconfiguration scenario poses no major problem and the order of the shutdown and the restart of components/VM is obvious. The other drawback, which seems obvious in the solution proposed in [20] is the systematic shutdown of all components and machines that support them. This solution seems a bit exaggerated and unrealistic. It would be wiser to stop only the components that should be stopped and let those which cannot be stopped run.

Our contribution in this paper is to model and propose an intelligent agent-based architecture, able to manage applications/services with integrated Cloud IoT while meeting performance criteria based on performance running time and power consumption. This work proposes an online reconfiguration algorithm that optimizes both the cost of storing and memory performance. For this reason, we opted for the modeling of a dependency graph whose vertices are the components/objects and edges are connections between these different components/objects, that can be executed on the same VM or different VMs. We also formalized a method of scheduling components/objects such that the start/stop components or objects is done in a consistent manner. As a result, we propose to use a weighted graph. On each arc, we add information that specifies if the shutdown of destination component imposes or not the shutdown of the source component. In this case, when a component needs to be stopped, stopping the component that precedes it will no longer be required, which allows a performance gain. To the authors knowledge, the first work that deals with the real-time reconfiguration of IoT and CC applications is what we propose in this paper which allows a performance gain in storage and power and for this reason we consider it as original work.

The remainder of the paper is organized as follows. Section 2 presents a brief overview of the application and reconfiguration of IoT and CC applications in manufacturing and embedded systems. In Section 3, we present our proposed solutions and implementation. Section 4 describes our case study and Section 5 discusses our experimental studies. Finally, Section 5 concludes this paper and gives avenues for future work.

II. State of the Art

Even though, the proposed approaches that offer cost models for storage devices are known to be particularly difficult, they have attracted the interest of many researchers. In [4], Kim et al. propose a model which optimizes the storage system based on the consumption of energy. Despite having applied this approach to several types of storage devices from the Hard Disk Drive (HDD) to the Solid-State Drive (SSD) [5][6] it seems to be difficult to apply the presented approach on VMs. Indeed, the VMs depend on several factors such as the type of the virtualization, the hypervisor and the VM reconfiguration (adding, deleting and modifying components/ objects) [4]. The model proposed by Kim et al. runs under devices which causes a limited memory, so our main challenge is to propose an intelligent agent-based approach for real-time reconfiguration of cloud applications that will guarantee the storage capacity as a performance factor.

Many other domains such as the transformation of the graph [7], meta-modelling [8], reconfiguration patterns [9], software architectures [10][11] have been addressed by many works in literature. In this sense, the works of Darwin [10] and Wright [11] help users to formally develop dynamic applications. The main characteristics of these formal models are the dynamic reconfiguration (adding or removing links) of component-based systems [17]. The protocol proposed by Salaun et al. [13] has the advantage of knowing the number of VMs, components and their relationships. However, the fact that this protocol is deployed in a cloud environment in a decentralized way does not guarantee the majority of applications that will require reconfiguration due to new requirements, scaling on demand or application techniques for recovery failures. Boyer et al. [14] propose a robust reconfiguration protocol to disconnect ports and to change the state of components. Despite the correctness of the protocol which is proven by the authors in their work, this approach has the inconvenient that all components are hosted on a same VM. In the same sense, a single centralized manager is used to ensure the steps of the reconfiguration protocol.

Other architectures are proposed in the literature and are based on a life cycle model that consists of basic states with a direct implication on the VMs and the Infrastructure As A Service (IAAS) layer. This framework supplies cloud services such as those proposed in Slim [5] and Claudia [11]. These frameworks take into account the configuration and infrastructure establishment on demand through the process of service deployment.

However, none of the existing works presents a practical model to reconfigure the cloud applications at run-time by using a modeling formalism, a graph of dependencies and the use of intelligent agents.

III. Proposed Contribution

In this section, we present our proposed contribution implemented by a multi-agent architecture.

A. *Motivation*

In CC, we encounter two major problems: computing and storage, and what interests us most is storage. In this case, we have the memory allocation and the devices status (free or busy). In this paper, two aspects will be dealt with consequences that are storage and memory performance factor where the goal of our contribution is to minimize the power consumption and the computing time. For storage, there are many works addressing these three methods: Deduplication, Compression (dealing with files) and Snapshot (treating devices while ensuring safety).

According to the work of Meyer and al. [22], the deduplication method is used to remove duplicated files. In this case, this method provides a gain of space of, e.g., X% and if we combine it with the compression method that is used to compress the files after deduplicating them, then it will guarantee a Y% gain. To deal with the problem of memory capacity, we will use the method of reconfiguration by proposing the use of a dependency graph to show the relationships between objects or components in VMs, with a weight of all dependencies (arcs) that predict the starting or not of these components and ultimately ensure the order of scheduling of these components or objects with topological sort. In this case, we must be able to propose a formal solution that ensures that the order to stop and start the components/objects or VMs must be coherent. We have to construct a graph of dependencies between components. This graph G = (V, E) is oriented where the vertices (V) are the components and arcs (E) are pairs of components (C_i, C_j). The starting, stopping, or removing of a component C_j should depend on the component C_i.

To construct the sequence diagram, we proceed to the topological sort of the dependency graph. The order of nodes indicates the coherent order of stopping or starting components or objects. The topological sort imposes Depth First Search (DFS) of the dependency graph. Then, we propose the use of a weighted graph G = (V, E, c) where c is the cost function defined by: c : $E \Longrightarrow \{0, 1\}$, where for each arc (C_i, C_j) we associate 1 the stopping of C_j requires stopping C_i and 0 otherwise. For each arc of the dependency graph, we add an information that specifies if the shutdown of destination component of this arc imposes or not the shutdown of the source component. In this case, when a component needs to be stopped, stopping the component that precedes will no longer be required, which allows a performance gain in memory. We take an example of n VMs composed of n components as shown in Fig. 1. Then, we apply the graph of dependencies,

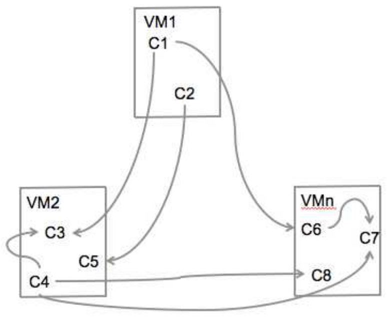

Figure 1. Example of components running in many VMs.

presented in Fig. 2. After that, we apply the second method of the topological sort as shown in Fig. 3.

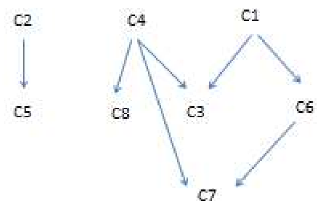

Figure 2. The graph of dependencies.

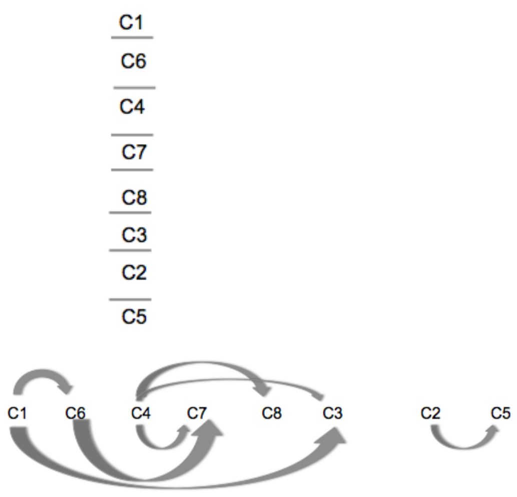

Figure 3. The topological sort.

B. Multi-Agent Architecture

To handle all possible types of reconfigurations, predictable or unpredictable, we propose an approach based on an intelligent agent-based architecture for real-time reconfiguration of CC and IoT applications. Our architecture is based on a hardware part composed of objects/components, a software part composed of VMs and the third part is cloud services including IAAS, Platform As A Service (PAAS) and Software As A Service (SAAS). We define five agents managing reconfiguration scenarios, as follows:

1) Hypervisor Agent (AH) in charge of creating VMs and controlling the cloud manager agent,
2) Cloud Manager Agent (ACM) in charge of the reconfiguration process and monitoring the status of deployed VMs, by controlling and assuring the interactions between the rest of the agents (the decisive agent, the evaluation agent and the executive agent),
3) EValuation Agent (AEV) which subsequently sends the request of reconfiguration operation to the executive agent and indicates success or failure against the following performance criteria: minimized energy criterion, maintenance and storage.
4) Decisive Agent (AD) according to the result of AEV, the AD decides to reconfigure again or not.
5) EXecutive Agent(AEX) runs components or operations in each VM and controls the relationships between them.

The proposed architecture of CC based on intelligent agents is shown in Fig. 4. Our architecture of IoT based on CC can be

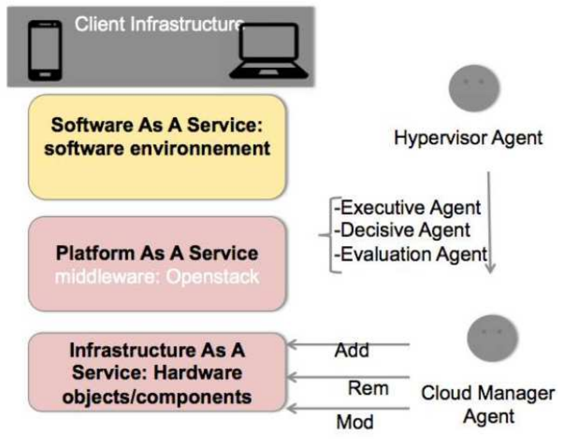

Figure 4. Architecture of cloud computing based on intelligent agent.

illustrated as shown in Fig. 5, where we take a smart hospital as a use case.

Smart Hospital

Figure 5. Architecture of IoT based on CC.

C. Formalization

We describe four dimensions of reconfigurations at run-time: reconfiguration of VMs, reconfiguration of objects or components, reconfiguration of relations (bindings) and reconfiguration of environment. Also, we consider the following reconfiguration operations: instantiation or destruction of VMs, addition or removal of a component to/from an existing VM, and addition or removal of liaisons. Now, let **Sys** be a distributed reconfigurable system composed of n VMs noted VM_n, where each VM is composed of *m* objects/components denoted C_m. We denote **NbCp_i** the number of components in each VM. Let A_{rs} be the agents to handle heterogeneous reconfiguration scenarios with **rs = [H,EX,EV,D,CM]** (rs: reconfiguration scenario). Let **E** be a finite set of states of each VM as follows: Active, Inactive or Suspended where **E = [A, I, S]**. Let **R** be the different reconfiguration operations: Addition, Removal, or Modification, where **R = [Add, Rem, Mod]**. Given the following matrix of size (n, 5) which defines scenarios that can be applied simultaneously by the different agents where each line corresponds to a reconfiguration scenario and the columns correspond to the VM_i, E, R, A_{rs}, and $NbCp\_i$ as described in Table I: We denote in the following

TABLE I. ARCHITECTURE DESCRIPTION

VM_i	E	R	A_{rs}	NbCp_i
VM_1	I	Mod/Add	A_{ex}	10
VM_2	A	Mod	A_{ev}	2
..	..	..	..	..

by: (i) $Instan^{AH}_{VM_n}$: a reconfiguration scenario applied by AH, (ii) $Reconf^{ACM}_{VM_n}$: a reconfiguration scenario applied by ACM, (iii) $Reconf^{AEX}_{VM_n}$: a reconfiguration scenario applied by AEX, (iv) $Reconf^{AEV}_{VM_n}$: a reconfiguration scenario applied by AEV, (v) $Reconf^{AD}_{VM_n}$: a reconfiguration scenario applied by AD.

Let Reconf be: $Reconf^{A_{rs},Cond}_{VM_i,C_j}$: a reconfiguration scenario applied by agents under two conditions: the finite state E and the different reconfiguration operations R, where i = [1, n] and j = [1, m]. A priority level must be defined between the different agents given as follows: (i) A_H: its priority is 1; (ii) A_{CM}: its priority is 2; (iii) A_{EV}: its priority is 3; (iv) A_D: its priority is 4; (v) A_{EX}: its priority is 5; We also define INT a set of interactions, where we distinguish two phases shown in Fig. 6: (i) Trigger phase: an agent which decides what direct interaction to launch and with which agent to interact, (ii) Interaction resolution phase: when an interaction is instantiated, to collectively choose an action from those proposed by the interaction, this action will be called the result of the interaction and its implementation will change the state of the overall system. For this reason, we must also define a transition function **Tr** between different VMs and that we characterize as follow:
If Sending **Then** Tr(VM_n, VM_{n-1}) = 1
Else Tr(VM_n, VM_{n-1}) = 0. An example is following:
1st step: $Instan^{AH}_{VM_i}$
2nd step: AEX may apply the following reconfiguration scenario :
$Reconf^{AEX,Cond}_{VM_i,C_j}$ where **Cond = [A, Rem]**, A corresponds to the state machine (Active) and **Rem** is the removal reconfiguration operation on the corresponding VM_i of the component C_j. We have to associate the following interactions between the different agents involved in the reconfiguration operation as follows:
$Int_{AEV\to AD}$. $Int_{AD\to AEX}$ if Tr(VM_i, VM_{i-1}) = 1.
3rd step: C_i = 0 (component is stopped) *Case 1:* $Reconf^{AEX,RemRi}_{VM_i,C_j}$: is a reconfiguration scenario of links where Rem_{R_i}: is the removal of links R_i. *Case 2:* $Reconf^{AEX,RemCj}_{VM_i,C_j}$, where Rem_{C_j} is the removal of the component C_j.
4th Step: C_j = 1 (component is active) *Case 1* : $Reconf^{AEX,cond}_{VM_i,C_j}$, where Cond = [A, Add]. *Case 2* : $Reconf^{AEX,AddR'i}_{VM_i,C_j}$.

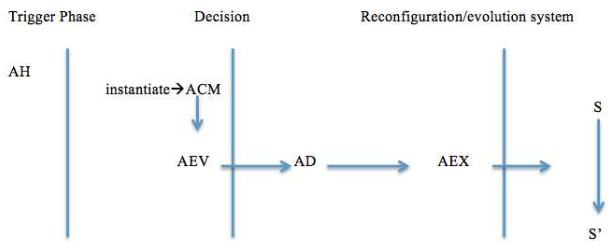

Figure 6. Interaction between agents.

```
Data:
Result:
1 begin
2     $date$ = 0; stack.init(); foreach $x \in S$ do
3         c[$x$] = WHITE;
4         p[$x$] = NULL;
5     foreach $x \in S$ do
6         if $c[x]$ = WHITE then
7             DFS($x$);
```

Figure 7. Topological sort algorithm.

D. Implementation

In this section, based on the proposed work [25], we implement the DFS algorithm which models our protocol described by the topological sort present in Fig. 7. The strategy of the DFS is to search in depth in the graph whenever possible. The path of a graph ends when all nodes have been visited. DFS will process the vertices first deep and then wide. After processing a vertex it recursively processes all of its descendants. The complexity of this algorithm shown in Fig. 8 is: T(n) = O(n + m), where n is the number of vertices (components) and m is the number of edges (liaisons between components).

IV. CASE STUDY

In this section, we present an evaluation of our proposed contribution by a case study in the healthcare filed.

A. Presentation

Healthcare is an example where IoT technologies are used to accelerate and coordinate management of medical

```
Data: $x$ : a node
Result:
begin
    c[$x$] = GREY;
    $date$ = $date$ + 1;
    d[$x$] = date;
    foreach $y \in a[x]$ do
        if $c[y]$ = WHITE then
            p[$y$] = $x$;
            DFS($y$)
    c[$x$] = BLACK;
    date = date + 1;
    stack.push($x$);
    f[$x$] = date;
```

Figure 8. DFS algorithm.

information and patient care. Our use case is composed of:
(i) Different services (cardiology, pediatric, etc.).
(ii) For each service, there are different applications. Some examples include the pacemaker which is in charge of the measurement and regulation of the patients heart beats, an electronic bracelet to track Alzheimers patients, etc.

Our proposed architecture of this use case is a private cloud where we modeled every service by a VM and each application is modeled by objects/components, as shown in Fig. 9. Each object (e.g. electronic bracelet) has an agent that controls the application composed of various agents to be distributed. These agents interact with asynchronous messages to exchange necessary information for starting/stopping the component. Each application has two FIFO buffers, one for incoming messages and one for outgoing messages. These applications interact together in a point to point mode (no broadcast or multi-way communication). Based on the case study discussed above, a simulation and a performance evaluation will be presented later.

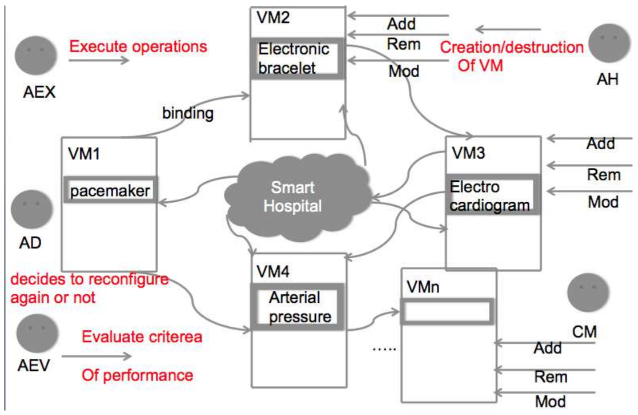

Figure 9. Architecture of smart hospital (Private Cloud).

We present a reconfiguration scenario shown in Fig. 10 and illustrated by the following steps:
(i) **1st step**: AH instantiates all VMs and controls the ACM.
(ii) **2nd step**: AEV failures on the balance of storage performance, as a consequence, AD demands to the AEX to reconfigure again. Then, AEX adds the required liaisons between the VMs.
(iii) **3rd step**: AEX removes the component C_j. (1) *Case 1:* AEX sends messages to VM2 for the removal of R_i. (2) *Case 2:* AEX sends messages to VM1 for the removal of R_1. (3) *Case 3:* AEX stops the execution of component C_1 and removes R_1. (4) *Case 4:* AEX stops the execution of component C_2 and removes R_2. (5) *Case 5:* AEX stops the execution of component C_j.
(iv) **4th step**: (1) *Case 1:* AEX adds a new C'_j. (2) *Case 2:* AEX executes the component C'_j. (3) *Case 3:* AEX adds a new relation R'_2. (4) *Case 4:* AEX executes the component C_2. (5) *Case 5:* AEX adds a new liaison R'_1. (6) *Case 6:* AEX executes the component C_1.

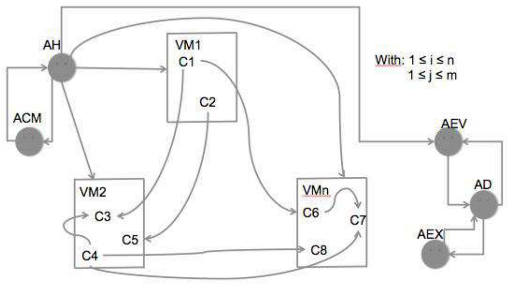

Figure 10. A reconfiguration scenario.

B. Application

In this section, we discuss the performance of our original proposed approach by simulation tests and we discuss its efficiency and effectiveness. We assume an environment of cluster with 27 heterogeneous physical machines (PMs) where three VMs are hosted. This environment includes 9 LENOVO hosts with AMD Athlon(tm) 64 X2 3600+ 1.9GHz, 9 DELL hosts with Intel(R) Core(TM)2 Duo 2.83GHz and 9 DELL machines with Intel(R) Core(TM)2 Duo 2.33GHz. We are varying our test parameters from 50 to 300 components and applying the test for each value and every result is the average of 20 trials of the experiment. This paper just discusses computing time and power consumption for each online reconfiguration scenario. Note that how to choose the most suitable values for each experimentation test is beyond the scope of this paper. Each type of the three VMs is simulated where each type of applications deployed in is referred to TPC-W benchmark [23].

C. Evaluation of Performance

In this section, we evaluate the efficiency and effectiveness of our online reconfiguration of IoT and CC applications approach. In Fig. 11, we show our proposed approach takes less computing time than the computing time required in [26]. Also, if we consider the A_{rs} parameter as a modification operation and the other three parameters (E, R and $NbCp\_i$) are assigned to fixed values, the computing time of our proposed approach is almost linearly increasing with the increase of interactions, which confirms that our approach has high scalability. Also, our online reconfiguration conserves the power consumption by up to 20% compared to the related approach of Tang et al. [27], as shown in Fig. 12.

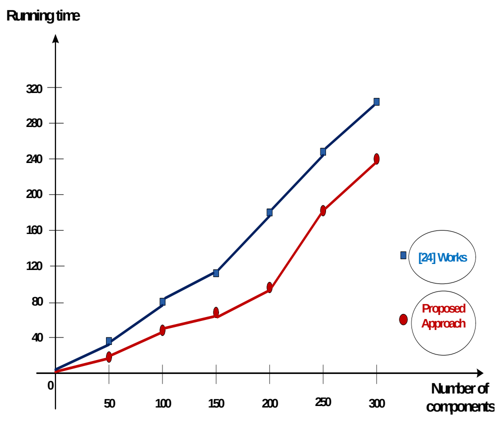

Figure 11. Running time evolution.

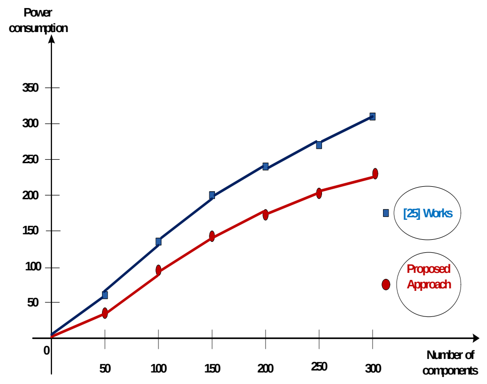

Figure 12. Power consumption evolution.

V. Conclusion and Future Work

This paper dealt with the performance criteria based on computing time and power consumption. We proposed an intelligent agent-based real-time reconfiguration able to manage applications in cloud and modeling formalism with algorithm to optimize these criteria. Our work assumes different applications deployed on VMs dependent on each other which is valid in reality and especially for current applications hosted in CC data centers and IoT applications. In our future work, we will continue our research taking into account real-time constraints that will occur in the capture of data from the sensors, storage capacity and security performance.

References

[1] N. Khanghahi and R. Ravanmehr, "Cloud Computing Performance Evaluation: Issues and Challenges". IJCCSA, vol.3, no.5, pp. 29-41, 2013.

[2] B. de Bruini and L. Floridi, "The Ethics of Cloud Computing". Sci. Eng. Ethics, pp. 1-19, 2016.

[3] I. Malhotra and J. Bakal, "A Survey and Comparative Study of Data Deduplication Techniques". IEEE Inter. Conf. on Pervasive Computing (ICPC), pp. 1-5, 2015.

[4] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam, "Hybridstore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs". in IEEE 19th Inter. Symp. on MASCOTS'11, pp. 227-236, Singapore, 2011.

[5] Z. Li, A. Mukker, and E. Zadok, "On the Importance of Evaluating Storage Systems Costs". 6th USENIX Workshop on HotStorage'14, pp. 6-6, Philadelphia, 2014.

[6] Y. Li and D.D.E. Long, "Which Storage Device is the Greenest? Modeling the Energy Cost of I/O Workloads". IEEE 22nd Inter. Symp. on MASCOTS'14, pp. 100-105, Paris, 2014.

[7] N. Aguirre and T. Maibaum, "A Logical Basis for the Specification of Reconfigurable Component-Based Systems". In Proc. of FASE03, vol. 2621 of LNCS, pp. 37-51. Springer, 2003.

[8] A. Ketfi and N. Belkhatir, "A Metamodel-Based Approach for the Dynamic Reconfiguration of Component-Based Software". In Proc. of ICSR04, vol. 3107 of LNCS, pp. 264-273. Springer, 2004.

[9] T. Bures, P. Hnetynka, and F. Plasil, "SOFA 2.0: Balancing Advanced Features in a Hierarchical Component Model". In Proc. of SERA06, pp. 40-48. IEEE Comp. Society, 2006.

[10] J. Magee and J. Kramer, "Dynamic Structure in Software Architectures". In Proc. of the 4th ACM SIGSOFT Symp. on Found. of Soft Eng. SIGSOFT96, pp. 3-14, 1996.

[11] J. Magee, J. Kramer, and D. Giannakopoulou, "Behaviour Analysis of Software Architectures". In Proc. of WICSA199, vol. 12 of IFIP Conf. Proc., pp. 35-49, 1999.

[12] R. Allen, R. Douence, and D. Garlan, "Specifying and Analyzing Dynamic Software Architectures". In Proc. of FASE98, vol. 1382 of LNCS, pp. 21-37. Springer, 1998.

[13] G. Salaun, X. Etchevers, N. De Palma, F. Boyer, and T. Coupaye, "Verification of a Self-configuration Protocol for Distributed Applications in the Cloud". In SAC12 Proceedings of the 27th Annual ACM Symp. on App. Comp., pp. 1278-1283, ACM Press, 2012.

[14] F. Boyer, O. Gruber, and G. Salaun, "Specifying and Verifying the Synergy Reconfiguration Protocol with LOTOS NT and CADP". In Proc. of FM11, vol. 6664 of LNCS, pp. 103-117, Springer, 2011.

[15] L. Srivastava, "The Internet of Things-Back to the Future", 2012, URL: http://www.youtube.com/watch?V=CJdNq.7uSdd.Mandfeature=related, [accessed 01-08-2016].

[16] "Green Goose", 2012, URL: http://www.greengoose.com, [accessed 04-08-2016].

[17] F. Boyer, O. Gruber, and D. Pous, "Robust Reconfigurations of Component Assemblies". In Proc. of ICSE13, pp. 13-22, IEEE/ACM, 2013.

[18] J. Kirschnick, J.M. Alcaraz-Calero, L. Wilcock, and N. Edwards, "Towards an Architecture for the Automated Provisioning of Cloud Services". In Proc. of ICSE13, pp. 13-22. IEEE Commun. Mag. 48, 124-131, 2010.

[19] L.R. Merino et al., "From Infrastructure Delivery to Service Management in Clouds". Future Generation Comput. Syst. 26, pp. 1226-1240, 2010.

[20] F. Duran and G. Salaun, "Robust Reconfiguration of Cloud Applications". The 17th Inter. ACM Sigsoft Symposium on CBSE'14, pp. 179-184, France 2014.

[21] P.P. Kumar, A.R. Reddy, and A. Rupa, "Snapshot Based Virtualization Mechanism for Cloud Computing". IJCSI, vol. 9, Issue 5, pp. 226-231, 2012.

[22] D.T. Meyer and W.J. Bolosky, "A Study of Practical Deduplication". ACM Trans. on Storage, vol. 7, no 4, Article 14, pp. 14:1-14:20, 2012.

[23] D.A. Menasc, "TPC-W: A Benchmark for E-commerce". IEEE Internet Computing, vol. 6, no 3: pp. 83-87, 2002.

[24] S.R. Kodituwakku and U.S. Amarasinghe, "Comparison of Lossless Data Compression Algorithms for Text Data". Indian Journal of Computer Science and Engineering, vol. 1, no 4, pp. 416-425, 2014.

[25] "Depth-First Search (DFS)", 2002, URL: http://www.cs.toronto.edu/~heap/270F02/node36.html [accessed: 06-08-2016].

[26] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments via Lookahead Control". Journal of Cluster Computing, vol. 12, no 1: pp. 1-15, 2009.

[27] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers". In Proc. of the 16th WWW'07, pp. 331-340, NY USA 2007.