Desktop Cloud Systems: Offering a Dependable Service

Carlos E. Gómez* Harold E. Castro† and Carlos A. Varela‡
*†Universidad de Los Andes - Bogotá, Colombia; ‡Rensselaer Polytechnic Institute - Troy New York, USA;
*Universidad del Quindío - Armenia, Colombia

Description. Desktop Cloud (DC) is a platform to provide cloud services running on desktop computers. A DC takes advantage of the idle computational resources usually in computer rooms, when the users are doing their daily activities. A DC manages the resources to execute functional virtual machines (VMs), with their operating systems and applications, without interfering in the activities carried out by end users. A DC can provide essential cloud services mentioned in [1]. UnaCloud [2] is our implementation of a DC, which supports the computational needs of scientific research works in an academic environment. UnaCloud is a best effort service. It only records some information regarding faults in VM deployments, but it does not use that information to make decisions about new deployments. It is the user's responsibility to track VMs that are running and restore them manually in case of fault. To have a dependable service is highly attractive for any cloud provider [3]. A dependable service would be useful to extend the services of a DC in general and UnaCloud in particular, so the range of applications that can be executed will be increased. For example, if the service were dependable, it would be possible to run a distributed system consisting of applications that communicate with each other, and finalize normally even if the infrastructure fails. According to Avizienis, et al. [4], dependability is the ability of a system to avoid faults that are more frequent and severe than acceptable. They say that dependability encompasses five attributes: availability, reliability, safety, integrity and maintainability. However, in [5], authors point out that dependability is a complex concept in which attributes that are chosen in a subjective manner by stakeholders are combined according to their particular requirements, and it covers the knowledge of faults, their evaluation, prediction, measurement and control. This doctoral research consists of analyzing broadly the faults that may occur in the normal operation of a DC, using UnaCloud as a testbed for our experiments. To achieve this, it is necessary to understand the behavior of the infrastructure and identify possible faults, obtain statistics for decision making about resource allocation (VMs to hosts), and have a mechanism that allows to the applications running on a DC to continue the execution of the system until finalize normally despite the faults that may occur in the infrastructure. Progress in a solution to facilitate the completion of a deployment of VMs on a DC against faults, is presented in this poster, including the results of new tests after the publication in SBAC-PAD 2017 [6].

Problem. As mentioned in the previous section, in order to offer a dependable service, we want to make a broad analysis regarding the faults of a DC. However, in this poster we focus on proposing a global snapshot solution as a mechanism to deal with infrastructure faults that interrupt the execution of applications belonging to a distributed system. Obtaining a global snapshot of a distributed system is not an easy task [7]. A global snapshot of a distributed system running on a DC, composed of applications that communicate with each other, is the set of local states of the VMs, along with a mechanism that allows the distributed system to preserve the communications in progress. A global snapshot is consistent if it allows users to resume the system execution without losing or duplicating messages. Messages sent before taking a local state and that do not have reached their destination, that is, those that have been left in transit, belong to the global snapshot. Then, it is necessary to guarantee that after resuming the execution from the global snapshot, these messages continue to their destination without altering the integrity of the applications. In this sense, we find that: 1) Hypervisors provide a function to obtain snapshots of individual VMs, but they do not consider groups of VMs running distributed systems. 2) It can not be guaranteed that the local snapshots of the VMs are taken at the same time. 3) After recovering the execution of the applications that run in the VMs, from a global snapshot, the successful reestablishment of communications can not be guaranteed. The main requirement of our solution is to obtain a consistent global snapshot that maintains the semantics of the system without modifying applications or hypervisors.

Methodology. We have taken Kangarlou's work [3] as a reference, in which VNsnap project depends on a virtual network called VIOLIN with VIOLIN switches running as VMs on each host. As a result, VNsnap is strongly coupled to its virtual network infrastructure. VNsnap uses additional infrastructure to support its operation since it requires VMs to implement its switches. Regarding the performance of the platform, all communications between two VMs pass through the VIOLIN switches, which causes a single point of failure and increases the latency in its respective host. As this solution uses UDP tunnels to send and receive frames passing going from one host to another, the reliability can be affected. Moreover, an overhead is caused due to the use of additional layers in the network stack, in the encapsulation process. We propose to obtain a consistent global snapshot for general distributed systems running on VMs that maintains the semantics of the system based on the algorithm used by VNsnap, in the context of UnaCloud. Our proposal emerges as an answer to the following two questions: How to preserve communications to obtain a consistent global snapshot? and if snapshots can not be taken at the same time, what is the appropriate time to do it? In our solution we will use a modification of the algorithm used by VNsnap to obtain a global snapshot adapted to UnaCloud. To preserve communications we will use the normal

behavior of the transport layer protocols (TCP and UDP). In addition, we differentiate the application messages that are part of a snapshot in each node of the distributed system from those that do not belong to a local snapshot at any given time. To do this, we use a mark (status information) that is placed on the outgoing datagrams and this mark is interpreted in the incoming datagrams of the receiver. In this way, we prevent that a message that is sent after taking the local snapshot in the sender (message that is not part of the local snapshot) can reach its destination before the receiver takes its local snapshot (the message would be included in the local snapshot). Regarding to the time in which the local snapshots are taken for each participant, we use a simple coordination protocol.

Implementation. Our solution is based on the snapshot operation offered by hypervisors and is characterized by being simple, it has a low impact on the performance of the distributed system to which it obtains global snapshots; does not require new infrastructure elements; and it does not affect the reliability of the applications. We take global snapshots, manage the generated files and allow the distributed system to resume its execution on the same or different hosts. To preserve communications, our solution is based on: 1) The TCP reliability mechanisms (if applicable); 2) Coloring of packets in the nodes (in the network layer); and 3) Filtering of network packets, also in the network layer. In addition, we designed a simple one-phase coordination protocol. If the distributed system running on the DC uses TCP as the transport layer protocol, sequence numbers, acknowledgments, timers, and retransmission of segments are necessary but not enough. It is possible the case of a system in which a message is sent after the sender has taken its local snapshot and the message arrives before the receiver has taken its local snapshot. To prevent it, our solution colors outgoing datagrams and filters in incoming datagrams.

Abstract of Results. We are interested in determining the variation of the time needed to obtain the local snapshots in several hosts, as well as to know the time needed to obtain a global snapshot. Additionally, we measured the overhead for executing the global snapshot protocol on the distributed system used for testing. We tested the performance of the solution using UnaCloud for the administration of the VMs, executing deployments with a variable number of VMs and hosts. A distributed system to run on the VMs, were developed. The nodes form a ring and exchange a numbered token a certain number of times. This application has a deterministic outcome, so it allows us to check easily if the execution was right or not, independently if during the process the global snapshot protocol were performed or not, or if the execution finished upon restoring the system from a global snapshot. We report the results considering the following metrics: global snapshot time, local snapshot time and overhead.

REFERENCES

[1] P. Mell and T. Grance. The NIST Definition of Cloud Computing. 2011.

[2] E. Rosales, M. Villamizar and H. Castro. UnaCloud: Opportunistic Cloud Computing Infrastructure as a Service. Paper presented at the CLOUD COMPUTING 2011 : The Second International Conference on Cloud Computing, GRIDs, and Virtualization., Rome, Italy.

[3] A. Kangarlou-Haghighi. Improving the reliability and performance of virtual cloud infrastructures. Doctoral Dissertation. Purdue University. 2011.

[4] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. IEEE transactions on dependable and secure computing, 1(1), 11-33. 2004.

[5] D. Prasad, J. McDermic and I. Wand. Dependability Terminology: Similarities and Differences. In Computer Assurance, 1995. COMPASS'95. Systems Integrity, Software Safety and Process Security. Proceedings of the Tenth Annual Conference on (pp. 213-221). IEEE.

[6] C Gómez, H Castro and C Varela. Global Snapshot of a Distributed System Running on Virtual Machines. *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) 2017*, Campinas, Brazil.

[7] A Kshemkalyani and M Singhal. Distributed computing: principles, algorithms, and systems. *Cambridge University Press*, 2008.

Carlos E. Gómez has been a University Professor at the Systems and Computing Engineering Department at Universidad del Quindío for 16 years. Carlos earned his M.Sc. in Engineering from Universidad de los Andes in 2007, and he is doing an Engineering Ph.D. at the same university since 2014. He was at the Rensselaer Polytechnic Institute - Worldwide Computing Laboratory as an intern for eight months between 2016 and 2017.