Executing and Pausing Distributed Applications Running on Desktop Clouds by Global Snapshots

Ejecutando y Pausando Aplicaciones Distribuidas Corriendo sobre Desktop Clouds Mediante Snapshots Globales

Carlos E. Gómez[1], Jaime Chavarriaga[2], David C. Bonilla[3], Harold E. Castro[4]

Gómez, C. E; Chavarriaga, J; Bonilla, D. C; Castro, H. E. Executing and Pausing Distributed Applications Running on Desktop Clouds by Global Snapshots. *Tecnología en Marcha*. Edición especial 2020. 6th Latin America High Performance Computing Conference (CARLA). Pág 44-48.

https://doi.org/10.18845/tm.v33i5.5074

1 Systems and Computing Engineering Department Universidad de los Andes, Bogotá, Colombia and Universidad del Quindío, Armenia, Colombia. E-mail: ce.gomez10@uniandes.edu.co.
https://orcid.org/0000-0002-5202-1167

2 Systems and Computing Engineering Department Universidad de los Andes, Bogotá, Colombia and Universidad del Quindío, Armenia, Colombia. E-mail: ja.chavarriaga908@uniandes.edu.co.
https://orcid.org/0000-0002-8372-667X

3 Systems and Computing Engineering Department Universidad de los Andes, Bogotá, Colombia and Universidad del Quindío, Armenia, Colombia. E-mail: dc.bonilla10@uniandes.edu.co.
https://orcid.org/0000-0002-3834-4736

4 Systems and Computing Engineering Department Universidad de los Andes, Bogotá, Colombia and Universidad del Quindío, Armenia, Colombia. E-mail: hcastro@uniandes.edu.co.
https://orcid.org/0000-0002-7586-9419

Keywords

Reliability; Fault tolerance; Checkpointing; Global snapshot; Desktop clouds.

Abstract

Desktop Clouds rely on volatile computing resources. For instance, platforms such as cuCloud and UnaCloud run scientific applications in virtual machines exploiting idle resources harvested in computer labs. Regretfully, these resources can be claimed by users, turned off and faulted at any time. The application running on these platforms suffer interference and interruptions that do not occur in dedicated platforms. We have been researching how to deal with these interruptions to increase the platform reliability and support applications running for large periods of time. This paper describes an application of our Global Snapshot Protocol, which can be employed for executing and pausing distributed applications running on desktop clouds. We found that, in these environments, the number of failures caused by desktop users is greater than the caused by hardware and communications. There, when a distributed system running in the virtual machines of a desktop cloud is paused, it can be restored in the same desktops, and successfully finish the application execution.

Palabras clave

Confiabilidad; Tolerancia a fallas; Checkpointing; Snapshot Global; Desktop clouds.

Resumen

Los desktop clods dependen de recursos computacionales volátiles. Por ejemplo, plataformas como cuCloud y UnaCloud ejecutan aplicaciones científicas en máquinas virtuales que aprovechan recursos ociosos en salas de cómputo y laboratorios. Lamentablemente, estos recursos pueden ser reclamados por los usuarios, apagados o presentar fallas en cualquier momento. La aplicación que se ejecuta en estas plataformas sufre interferencias e interrupciones que no ocurren en plataformas dedicadas. Nosotros hemos estado investigando cómo enfrentar estas interrupciones para aumentar la confiabilidad de la plataforma y soportar aplicaciones que se ejecutan durante largos períodos de tiempo. Este artículo describe una aplicación de nuestro Protocolo de Snapshot Global, el cual puede emplearse para ejecutar y pausar aplicaciones distribuidas que se ejecutan en desktop clouds. Nosotros encontramos que, en estos entornos, la cantidad de fallas causadas por los usuarios de los computadores de escritorio es mayor que la causada por el hardware y las comunicaciones. Allí, cuando se detiene un sistema distribuido que se ejecuta en las máquinas virtuales de un desktop cloud, nosotros podemos reanudar la ejecución usando los mismos computadores y finalizar exitosamente la ejecución de las aplicaciones.

Introduction

Desktop clouds (DC) offer cloud computing services using idle resources on common desktop computers [1]. Typically, DCs offer infrastructure services (IaaS) such as running virtual machines (VMs) and virtual clusters. For instance, researchers can run Bag of Tasks (BoT) applications that divide the work in chunks and distribute them on virtual machines running on multiple desktops. In addition, they can use platforms such as cuCloud [2] and UnaCloud [3] to run clusters of customized virtual machines to run more complex distributed applications. All these platforms help researchers to run scientific applications at lower costs than using dedicated datacenters.

In a DC, computing resources are very volatile because the platform harvest idle resources from desktop computers. A desktop user may claim these resources, kill some processes or turn off the physical machines (PM). DCs are more susceptible to failures at runtime than other cloud platforms. For instance, when the users run BoT applications, these failures can halt the execution of some chunk of work. There, the platform can solve these interruptions by assigning the failed chunks to other virtual machine and starting again its processing. However, when DC users run distributed applications, these failures are hard to manage. Distributed applications, such as GROMACS and other MPI-based, fail if one of the nodes is not accessible. Any failure of a virtual machine may result in the lost of many hours of work. Existing platforms do not provide strategies nor tools to offer warranties for executing distributed applications during large periods of time.

We have been researching how to improve reliability on desktop clouds. In a previous paper, we proposed a global snapshot protocol and develop a software tool to obtain a consistent global snapshot for a general distributed system running on virtual machines [4]. The software tool maintains the semantics of the distributed system without modifying applications running on virtual machines or hypervisors. In this paper, we present a use case the global snapshot system that allows the desktop cloud user to pause the execution of the distributed system that runs on virtual machines and resume its execution. This use case is important, especially when we have distributed applications whose execution is carried out for long periods of time, for example, weeks or months, it may be necessary to pause its execution, without losing the work done so far.

Global Snapshot System

A Global Snapshot creates a checkpoint for a general distributed system, preserving the state of the participant nodes and the communications among them. Using these snapshots, it is possible to resume the execution of both the processing and the communications in progress.

In [4] we presented a Global Snapshot Protocol and a software tool that can be applied to save the state of collections of VMs running on multiple computers. Here the virtual machines communicate with each other using normal network devices instead of virtual networks and it is not possible to take “almost at the same time” snapshots of VMs and the communication channels. Creating a checkpoint requires to coordinate local snapshots in each node and provide means to resume the in-progress transmissions among them.

Our implementation is able to create snapshots of distributed systems on DCs without requiring modifications on the applications, the VMs or the hypervisors. The concepts used in the implementation are the following: TCP reliability mechanisms (if applicable); colors in the packages in the nodes, in the network layer; packet filtering, again, in the network layer and we adapt a simple coordination protocol to obtain a global snapshot at any time.

The protocol is an adaptation of the two-phase commit protocol to coordinate the participating desktops. The first phase consists of a previous verification in which the coordinator (one of the participants) consults each process if the VMs corresponding to the system are in execution. If all VMs are running, we go to the second phase, which consists of taking the global snapshot. Applying the concept of colors to identify datagrams, we use colorless datagrams when the global snapshot protocol is not in operation, that is, before starting and after finishing, and we assume that a global snapshot will not start before finish the previous one. In addition, the protocol uses the operating system running in the VMs to process the network communications by modifying control fields, marking the outgoing datagrams and filtering out some of the incoming ones. The process has been designed based on the expected behavior of the TCP and UDP protocols. In this way, a datagram sent that does not reach its destination, will be retransmitted after resuming

execution, if the transport layer protocol is TCP. On the contrary, if the protocol is UDP, the packet will not be retransmitted, as it happens in this type of applications.

On the other hand, the system has a resume protocol to coordinate the process when it is desired to continue the execution of a distributed system from a global snapshot. In this case, it cannot be guaranteed that when resuming the execution of the VMs from a global snapshot, they do so at the same time. Therefore, we create three-phase protocol, to coordinate the restoration of the system, looking for communications to be restored and that the system can successfully complete its execution.

The first phase is to verify that the physical machines, where VMs are hosted, are running. If any of the PMs is not ready, a timer is activated and after a timeout, the verification starts again. When all the PMs have answered to the coordinator, we will go to phase 2. In the second phase the restore point is established in a specific snapshot (typically the last one), although it could be any. The coordinator sends a message to the other participating processes. When all the PMs have answered the coordinator, we move on to phase 3. The third phase is similar to the second one. The coordinator sends the other participants a message and they give an answer. The coordinator determines that the resumption has been successful when he receives the response from all participants.

Executing and pausing distributed applications: Experience report

In our UnaCloud, we are running GROMACS processes that spent several days, using clusters of tens of physical machines. Although we can deploy clusters with hundreds of virtual machines, the applications may run slower on them because the use of non-dedicated network connections.

We perform multiple functional and performance tests to determine the effectiveness of the tool developed. We used a virtual image with the Ubuntu Server 16.04 operating system, which occupied 3.51 GB to implement virtual machines with 1 GB of RAM, 5 GB of virtual hard disk and 1 processing core. The virtual machines were run on physical machines with an Intel Core i7-4770 processor, 20 GB of RAM and 500 GB of hard disk. We use a computer lab with 20 desktop computers, connected via a 1GB Ethernet network. We use a GROMACS-MPI benchmark in the lab, running 1 VM on each PM and 2 processes for each node. The test took approximately 11 hours to complete, and we took snapshots every 1 hour.

It resumed from snapshot # 7 and from snapshot # 11 (the last one). In both cases, the test ended successfully.

It is very important consider that probability of failure in computer labs helps to decide to run and pause the distributed system because we found that, in our case, the number of failures caused by desktop users is greater than the caused by hardware and communications. The computer labs are upgraded every three or four years and the hardware typically do not fault. According to official statistics, there were only 11 incidents for faulty hardware during the last 5 years. Our studies about failures that occur in a DC [5] shows that desktop users are the main cause of problems. If necessary, we can use our global snapshot tool to pause system execution and resume it at the desired time, avoiding losing the work done so far.

References

[1] A. Alwabel, R. J.Walters, and G. B.Wills, “A View at Desktop Clouds,” in International Workshop on Emerging Software as a Service and Analytics (ESaaSA 2014), pp. 55–61, 2014.

[2] T. M. Mengistu, A. M. Alahmadi, Y. Alsenani, A. Albuali, and D. Che, “cucloud: Volunteer computing as a service (vcaas) system,” in International Conference on Cloud Computing, pp. 251–264, Springer, 2018.

[3] E. Rosales, H. Castro, and M. Villamizar, “UnaCloud: Opportunistic Cloud Computing Infrastructure as a Service,” Cloud Computing, pp. 187–194, 2011.

[4] C. E. Gómez, H. E. Castro, and C. A. Varela, “Global Snapshot of a Distributed System Running on Virtual Machines,” in 2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 169–176, IEEE, 2017.

[5] C. E. Gómez, J. Chavarriaga, and H. E. Castro, “Fault characterization and mitigation strategies in desktop cloud systems,” in Latin American High Performance Computing Conference, (Cham), pp. 322–335, Springer International Publishing, 2019.