Future Generation Computer Systems 34 (2014) 17–25

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Fault-tolerant virtual cluster experiments on federated sites using BonFIRE☆

A. Gómez[a,*], L.M. Carril[a], R. Valin[a], J.C. Mouriño[a], C. Cotelo[a,b]

[a] Supercomputing Centre of Galicia, Campus Vida, Avda. de Vigo S/N, 15705 Santiago de Compostela, Spain

[b] Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Campus Vida, 15782 Santiago de Compostela, Spain

HIGHLIGHTS

- A new proposal for a virtual cluster architecture with fault-tolerance for Cloud.
- A new Elasticity Engine that uses the application performance.
- Elasticity of virtual clusters using application performance monitoring.
- Experiment result about using elasticity to fulfill Specific Deadlines Objective.
- Fault-tolerant experiment results using BonFIRE's federated infrastructure.

ARTICLE INFO

Article history:
Received 17 December 2012
Received in revised form
22 September 2013
Accepted 3 December 2013
Available online 19 December 2013

Keywords:
Cloud
Federation
QoS
Monitoring
Elasticity
Fault-tolerance
HTC
Virtual cluster

ABSTRACT

The failure of Cloud sites and variability of performance of the virtual machines (VMs) in this environment are two issues that have to be taken into account by software providers. If they want to guarantee the return of the results on time to their customers, their virtual infrastructure must be designed to adapt itself to the new scenario. This is especially critical in compute intensive applications that execute on virtual clusters with a large number of VMs, because they can need hours or days to produce valid results. Changes in the performance could mean longer times to produce results and, probably, higher costs. Site failures usually force to restart from the beginning, losing many computing hours. In this paper we present a fault-tolerant virtual cluster architecture that can tackle with both issues in the context of compute intensive bag-of-tasks applications. It includes an Elasticity Engine that uses the application performance to decide about the enlargement or reduction of the virtual cluster to fulfill the expectations of the final users. The architecture has been tested in three experiments: execution of the application in a multi-site configuration which has shown that it is not suffering from any penalty because of its execution in a distributed environment; an experiment about Specific Deadline Objective where the Elasticity Engine takes decisions about the enlargement of the cluster with new VMs to end the simulation on time; and a fault-tolerance test where one part of a distributed virtual cluster is lost, restoring the application performance on the surviving Cloud site using recovering mechanisms and elasticity rules, without interruption of the service.

1. Introduction

Cloud providers aim to have a pervasive service for their customers as other utilities do. However, as the electric grid, they are not free of outages. In fact, in the summer of 2012, two of the biggest Cloud providers in the market suffered service interruptions in some areas [1–3]. As a consequence, some of their customers' services were down totally or partially. Because these service interruptions were at site level, the unique solution to mitigate the impact is either to deploy them on several sites of the same Cloud provider or even to use more than one provider. In both cases, a fault-tolerant architecture is mandatory. In addition to this issue, Software-as-a-Service providers which use these big Cloud infrastructures must tackle the possible variability in the application performance due to many causes, such as changes in the assigned hardware between deployments, or sharing the infrastructure with other customers. In fact, J. Schäd et al. [4] have observed a performance variance in Amazon EC2, even having two different levels of performance for the same configuration of their virtual infrastructure but measured on different times. In this case, to maintain the quality of the service, a possible strategy is to exploit the elasticity of the Cloud to self-adapt to this undesired variability. This

☆ The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007–2013) under grant agreement number 257386.

* Corresponding author. Tel.: +34 981569810.
E-mail addresses: agomez@cesga.es (A. Gómez), luis.rodriguez@kit.edu (L.M. Carril), raul@raulvalin.es (R. Valin), jmourino@cesga.es (J.C. Mouriño), carmen@cesga.es (C. Cotelo).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.027

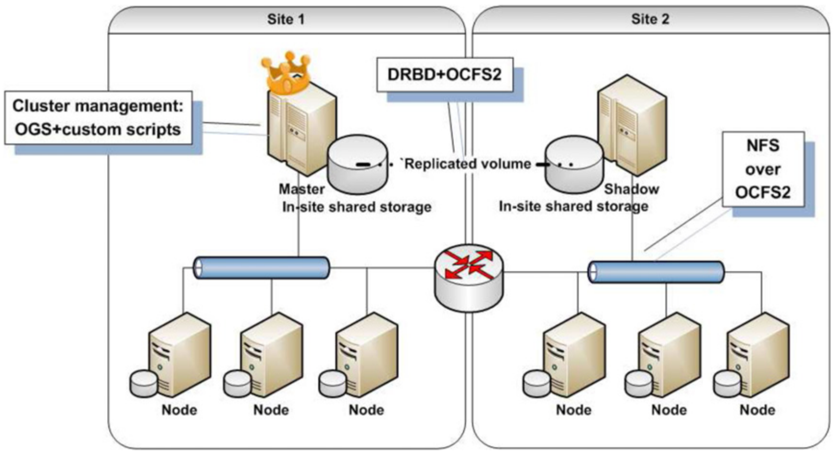

Fig. 1. Virtual cluster architecture. It comprises master and shadow nodes plus a set of computer elements. A shared replicated file system is created using OCFS2.

means using a key application performance indicator to trigger the enlargement of the backend virtual infrastructure when needed or for reducing it when it is idle to avoid unnecessary costs. A set of applications that can benefit from this Cloud horizontal elasticity is those that can be divided in several independent tasks, as Monte Carlo simulations or engineering parametric studies. For Software-as-a-Service providers of these applications it is important to return the results on time, with some level of guarantee, and with the lower possible costs. In some cases, they could have Service Level Agreements (SLA) with their customers to return the results before a limit hour. Examples of such services are the executions of ensembles of atmospheric numerical solutions for operational weather forecast, engineering simulations for product design optimization, or Monte Carlo simulations of clinical radiotherapy treatments.

In this paper we present a fault-tolerant virtual cluster architecture that can tackle with the changes in the performance and the site failures in the context of compute intensive bag-of-tasks applications. It includes an Elasticity Engine that uses the application performance to decide about the enlargement or reduction of the virtual cluster to fulfill the expectations of the final users. To check the design of the architecture, it has been tested in three experiments: execution of the application in a multi-site configuration; an experiment about Specific Deadline Objective where the Elasticity Engine takes decisions about the enlargement of the cluster with new VMs to end the simulation on time; and a fault-tolerance test where one part of a distributed virtual cluster is lost, restoring the application performance on the surviving Cloud site using recovering mechanisms and elasticity rules, without interruption of the service. The presented architecture is well suited to cases where one virtual cluster is allocated to a single execution of the application for one customer, without sharing it between customers and other applications, and is deployed on demand in one or several Cloud providers. It is completely autonomous, adapting itself to changes in the Cloud provider infrastructure.

This paper is divided in five sections. First of all, technical details about the proposed virtual cluster architecture, elasticity management using application performance, use case and experiment infrastructure are presented. The next section describes the experiments executed to check the proposed architecture. A brief review of related work is presented in Section 3. Finally, next two sections summarize the conclusions and describe future work.

2. Materials and methods

2.1. Virtual Cluster Architecture

The full virtual cluster (VC) architecture comprises two main nodes (“master” and “shadow”) and two sets of computing nodes (CE); each set is associated to one of the main nodes (see Fig. 1). So, the VC is split in two partitions which could be deployed in different locations. To build it, two different VM images have to be configured: one for the master and shadow nodes, and another one for the CEs. The dependencies between nodes, the location of each partition, and other technical requirements are automatically self-configured during the final deployment.

The master node has four tasks: data server for CEs that are deployed in the same Cloud site, queue controller, application manager, and cluster configuration. The shadow node has another three tasks: data server for its clients, backup queue controller, backup application manager. The compute nodes have only a single task: execution of jobs.

The four components that the VC must have to perform the tasks described above are:

- Shared storage resource. It is needed to manage and synchronize the queue system, and to store all the data of the running processes. As an external shared file system is not commonly offered by Cloud providers and in a multi-cloud environment without federation maybe will not be available in the near future, a different solution is mandatory. Therefore, an empty volume is also created and attached to the main node of each site (master and shadow). The content of both volumes are replicated and synchronized. This is done combining DRBD (volume replication at block level) [5] and OCFS2 (Oracle Cluster File System) [6], which provide a fully replicated volume with dual-primary access (the two main nodes can write simultaneously). The initial synchronization can impose a big penalty in the VC setup. To minimize it, the shared volumes have been created previously at BonFIRE sites as DATABLOCKs and filled with zeros. When they are instantiated in each site, as both of them are empty, the synchronization is done quickly. CE access to the shared files system using NFS, thanks to OCFS2, is compatible with NFS exportation.
- Queue system. To distribute the parallel jobs among the nodes, Open Grid Scheduler (OGS) [7] is used. The master and shadow nodes are configured as OGS master and shadow daemons respectively (and both of them as submit and administration hosts); only CEs are execution hosts. Independently of the partition where the CE is included, it can communicate with the master (or the shadow when the first is not available) for queue operations and job distribution.
- Self-configuration. External intervention is undesirable after deployment request to setup the VC. So the cluster self-configures. The master detects the initial machines in the cluster and setups all the components. It also connects to the shadow using SSH to make its setup.
- Application software. It must be an application which can be divided in several tasks and can inform the virtual cluster about its