2014 IEEE International Conference on High Performance Computing and Communications (HPCC), 2014 IEEE 6th International Symposium on Cyberspace Safety and Security (CSS) and 2014 IEEE 11th International Conference on Embedded Software and Systems (ICESS)

Implementation of the KVM hypervisor on several cloud platforms: tuning the Apache CloudStack agent

F. Gomez-Folgar
A.J. Garcia-Loureiro
T.F. Pena
CITIUS. USC, Spain
Email: fernando.gomez.folgar@usc.es

J.I. Zablah
Sistema de Difusión de Radio y Televisión
Universidad Nacional Autónoma de Honduras
Tegucigalpa, Honduras

N. Seoane
Electronic Systems Design Centre
College of Engineering
Swansea University, UK

***Abstract*—The default KVM setup included in OpenNebula, Eucalyptus and Apache CloudStack could not perform well enough under certain circumstances, because it does not expose to the guest VMs some of the features included in the modern processors. In order to tune up the cloud platform for obtaining a better CPU performance for the virtualized software executed under the KVM hypervisor, we have proposed and tested an alternative KVM setup in Apache CloudStack. The setup provided could also be applied to other cloud platforms.**

I. Introduction

Cloud technologies [1]–[3] are arousing great interest because of the current increase on the demand by users of virtualization services and Virtual Machines (VMs). As a result, there is an increasing number of open-source solutions for building private, public and even hybrid Clouds that can make use of different hypervisors, including KVM [4]. However, the specific way in which the hypervisor is tuned can have important effects on the performance of the applications that will be executed in the VMs. For this reason, in this paper, we have analysed the KVM configuration included in three popular open-source cloud management platforms such as OpenNebula [5], Eucalyptus [6] and Apache CloudStack [7]. In the analysed platforms, KVM provides the VMs with a basic abstraction of the underlying CPU, hiding some advanced CPU features. One of the reasons of this limitation is to guarantee the guest compatibility with different hardware in heterogeneous environments. However, in a large number of situations, the cloud is deployed on an homogeneous set of hosts. In these cases, this limitation can severely reduce the performance of the guests systems. To overcome this situation, we have proposed a new configuration to allow the guest VM CPU to take advantage of the modern processor features. The proposed changes were implemented and tested in an Apache CloudStack infrastructure but could be applied to other cloud management platforms.

This paper is organized as follows. In section II we present the architecture and the main features of the three popular open-source cloud management platforms previously mentioned. Section III summarizes the modifications that have been performed on the Apache CloudStack agent. Section IV includes the performance test and analyses the obtained results. Finally, the main conclusions of the paper are drawn in the last section.

II. Open-source cloud management platforms

Currently, there have been important developments of open-source cloud management solutions with the release of several platforms such as OpenNebula, Eucalyptus and Apache CloudStack that provide different architectures and features. The main characteristics of these infrastructures are as follows:

A. *OpenNebula*

OpenNebula is an open-source software that allows building private, public and hybrid clouds. It adopts a classical cluster-like architecture with a front-end, a datastore, a network and a set of hypervisor-enabled nodes, where the VMs can be executed and managed. As we can see in Fig. 1, the main components are the front-end, the nodes, the datastore and the management daemon (ONED). The front-end executes the management daemon and cluster services. The nodes are hypervisor-enabled hosts that can execute VMs. The datastore can be any storage system, such as SAN or NAS servers, used to store disk images for VMs. OpenNebula employs at least two different physical networks: a service network and an instance network. The service network is used by the front-end to manage hosts and hypervisors, and to distribute image files. The instance network provides the network connectivity to the VMs.

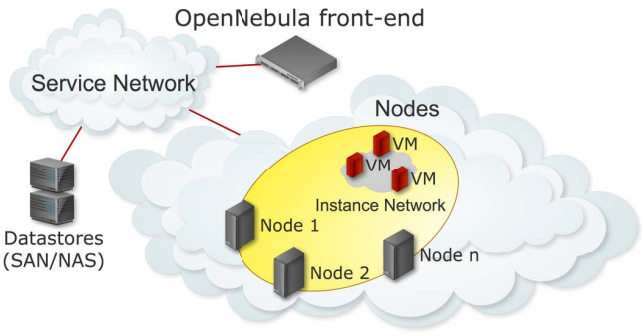

Fig. 1. OpenNebula architecture.

978-1-4799-6123-8/14 $31.00 © 2014 IEEE
DOI 10.1109/HPCC.2014.46

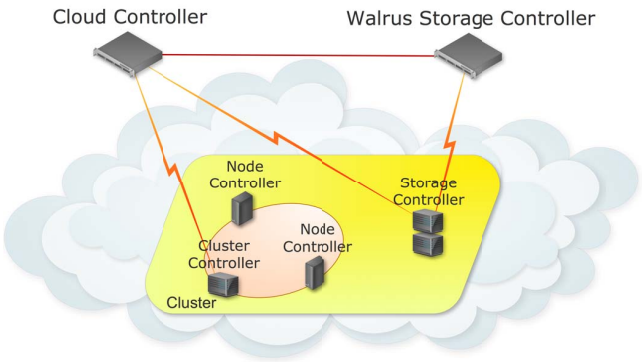

Fig. 2. Eucalyptus architecture.

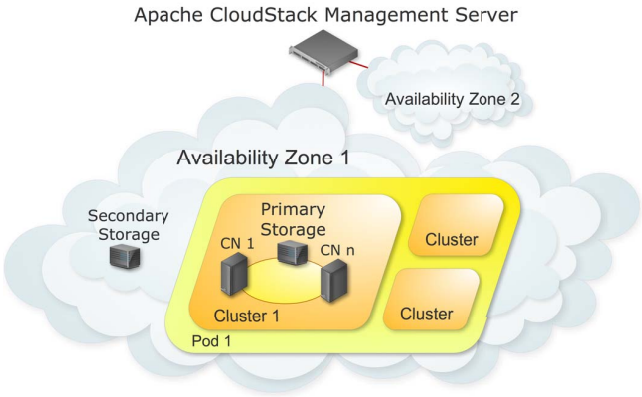

Fig. 3. Apache CloudStack architecture.

The hypervisors supported by OpenNebula are Xen, KVM and VMware ESX. This cloud platform can be managed through the Command Line Interface (CLI) or the SunStone web interface.

B. Eucalyptus

Eucalyptus is an open-source software that allows building private and hybrid clouds employing external resources. The Eucalyptus infrastructure, shown in Fig. 2, is mainly composed by five types of components: Cloud Controller (CLC), Cluster Controller (CC), Storage Controller (SC), Node Controller (NC) and Walrus Storage Controller (WS3). The CLC monitors the availability of resources in several components of the infrastructure, such as Nodes and CCs. WS3 implements the persistent data storage system that stores machine images and snapshots. The CC performs the tasks of planning and controlling resources at cluster level. The SC manages the Elastic Block Store at cluster level. The NC manages the life-cycle of the VM instances.

The hypervisors supported by this cloud platform are Xen and KVM. The Eucalyptus cloud can be managed through its web interface or by means of CLI interfaces.

C. Apache CloudStack

Apache CloudStack is an open-source cloud platform that allows building any type of cloud: private, public and hybrid. The Apache CloudStack software architecture is composed by seven types of components, as depicted in Fig. 3: Apache CloudStack Management Server, Availability Zone, Pod, Cluster, Compute Nodes (CNs), Primary Storage System and Secondary Storage System. The Apache CloudStack Management Server controls the cloud and the allocation of VMs. An Availability Zone is like a single datacenter that is composed by several Pods and a Secondary Storage System. A Pod is a rack of hardware including layer-2 switches and one or more clusters. A cluster is composed by several CNs that share the same hypervisor and the same Primary Storage System. CNs are hypervisor-enabled hosts managed by Apache CloudStack. The Primary Storage System is also associated with the Cluster and stores the root filesystem of guest VMs. The Secondary Storage System is associated with the Availability Zone and stores VM templates, ISO images and disk volume snapshots.

The hypervisors supported by Apache CloudStack are KVM, VMware vSpere and Citrix XenServer. Apache CloudStack supplies an API and a web interface that provides the complete management of the cloud for the administrators, including very interesting options like the the possibility of installing the OS of VMs using standard ISO images.

III. Tuning KVM configuration

A. Default KVM setup

The default KVM configuration employed by OpenNebula 4.4, Eucalyptus 3.4.2, and Apache CloudStack 4.0.1 provides the VMs with a basic abstraction of the underlying CPU that could not perform well enough under certain circumstances, because it can not take advantage of some features of the hosts processors. Currently, with the default configuration, the processor exposed into the VM is a QEMU Virtual CPU with the following flags or instruction sets:

```
fpu, de, pse, tsc, msr, pae, mce, cx8, apic, mtrr,
pge, mca, cmov, pse36, clflush, mmx, fxsr, sse, sse2,
syscall, nx, lm, rep_good, pni, cx16, hypervisor.
```

This limitation on the set of instructions of the VM processors makes the replacement of the host processor or the VM migration easier, guaranteeing the guest compatibility with the new host. This configuration is appropriate for maximizing the compatibility in a heterogeneous environment. However, guaranteeing this compatibility comes at a performance cost.

B. Tuning KVM in the Apache CloudStack agent

In the Apache CloudStack platform, each managed compute node must have an Apache CloudStack agent installed. This agent is the component responsible for managing the VMs and its configuration in the hypervisor-enabled nodes.

If all the Apache CloudStack enabled hosts have more CPU features than the exposed by the default Apache CloudStack KVM configuration, it is possible to tune up the configuration to expose the extra features to the guest VMs, and also retain the compatibility for the VM live migration.

Apache CloudStack, as OpenNebula and Eucalyptus, does not perform the management of the KVM hypervisor in a direct way, but they employ the libvirt virtualization API. Libvirt [8] is a toolkit that interacts with the virtualization capabilities

of recent versions of Linux, providing features to perform the management of the VMs.

Tuning the Apache CloudStack agent is necessary in order to adapt the libvirt VM definition to the parameters of the new configuration. In this case, the $LibvirtVMDef$ class is involved in the management of the configuration of the hypervisor on the compute nodes. The CPU model exposed into the VM can be managed using the CPU element in the libvirt VM definition file. The CPU element is the container that holds the description of the guest CPU. As the compute nodes we are employing match at least Intel Nehalem architecture, modifying the appropriate KVM configuration allows matching at least that architecture, exposing a set of new features to the VM by adding the required definition to the $LibvirtVMDef$ class.

```
<cpu mode="custom">
    <model fallback='allow'>Nehalem</model>
    <feature policy='optional' name='sse'/>
    <feature policy='optional' name='sse2'/>
    <feature policy='optional' name='sse4.1'/>
    <feature policy='optional' name='sse4.2'/>
    <feature policy='optional' name='ssse3'/>
    <feature policy='optional' name='pni'/>
</cpu>
```

With this new configuration, the VMs managed by Apache CloudStack will take advantage of the extra CPU features, allowing the applications to be executed into VMs to make use of the extra CPU features too. The virtual CPU is now exposed into the VMs as Intel Core i7 9xx (Nehalem Class Core i7) with new instructions, such as SSSE3, SSE4.1, and SSE4.2, including the following flags or instruction sets:

```
fpu, de, pse, tsc, msr, pae, mce, cx8, apic, mtrr,
pge, mca, cmov, pat, pse36, clflush, mmx, fxsr, sse,
sse2, syscall, nx, lm, constant_tsc, nopl, pni, cx16,
ssse3, sse4_1, sse4_2, x2apic, popcnt, hypervisor.
```

Both OpenNebula and Eucalyptus could be tuned in a similar way. In the case of OpenNebula, the component responsible of the management of the hypervisor is $LibVirtDriverKVM$. In the case of Eucalyptus, $libvirt\_tortura$ is the component responsible of the hypervisor configuration management.

IV. Results

In order to evaluate the performance of the VMs deployed with both the Apache CloudStack tuned agent (described in the previous section) and the default one, we have executed three applications in the VMs, employing 1, 2, and 4 VCPUs. We have used a cloud infrastructure based on Apache CloudStack 4.0.1, with KVM as the managed hypervisor in the CNs. In this infrastructure, the Primary Storage System and the Secondary Storage System are composed by 12 TB NAS in RAID 5. The CNs have Intel Core i7-2600@3.40 GHz processors with 8 GB of RAM and CentOS 6.3 64 bit as OS. The interconnection network of this cloud infrastructure is Gigabit Ethernet.

The applications selected for testing both Apache CloudStack agents were Intel Linpack [9] 11.1.2 version, HandBrake [10] 0.9.9 version, and 3D DD-DG, an in-house parallel three-dimensional drift-diffusion (D-D) semiconductor device simulator [11]. The VMs used have 7.0 GB of RAM and VCPUs ranging from 1 to 4. Intel Linpack and the 3D DD-DG semiconductor simulator were executed under CentOS 6.3 64 bit OS whereas HandBrake was executed in Ubuntu 13.10 64 bit OS, so to test two different environments.

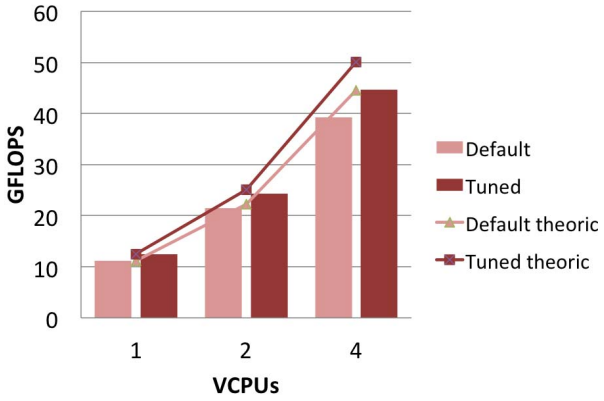

Fig. 4. Performance for Intel Linpack.

The Intel Linpack benchmark solves a dense system of linear equations and takes the amount of time that it employs to solve the system as a measure of the system's floating point computing power. In this case of study, the size of the systems to solve is 5,000 with alignment of 4KB.

HandBrake is an open source video transcoder than can process the most common multimedia files. In this case of study, we have processed a video provided by the UNAH TV [12], which is used in their TV broadcast system, and we have converted it from MTS format to mp4. The source video is stored in a MTS container that has two streams. The video stream is coded with AVC and has an anamorphic resolution of 1920x1080 pixels at a frame rate of 29.970 fps, employing Variable Bit Rate (VBR). It has 54,448 frames and a length of 30 min, 16 s and 748 ms. The audio stream is coded in PCM, employing Constant Bit Rate (CBR) at a sampling rate of 48.0 KHz at 16 bit. The size of MTS file is 1.98 GiB. The output video is stored in a mp4 container that has both the video and the audio streams. The output video stream is coded with AVC and has an anamorphic resolution of 1440x816 pixels at a frame rate of 29.970 fps employing VBR. The output audio stream is coded in AAC, employing VBR at a sampling rate of 48.0 KHz. The size of mp4 file is 253 MiB.

The 3D DD-DG simulator is a parallel three-dimensional drift-diffusion (D-D) semiconductor device simulator that includes quantum correction effects through the density gradient (DG) approach. The DD and DG equations are discretised using a finite element method on a unstructured tetrahedral mesh. The obtained set of equations is first decoupled, using Gummel's method, and then linearised, using Newton's method. The resulting linear systems of equations are solved in parallel on an arbitrary number of processors using MPI. To partition the solution of the linear systems to each of the available processors, domain decomposition methods based on the Additive Schwarz procedure are employed, which is highly parallel. This simulator has been employed to study the impact of different sources of variability on FinFET devices [11].

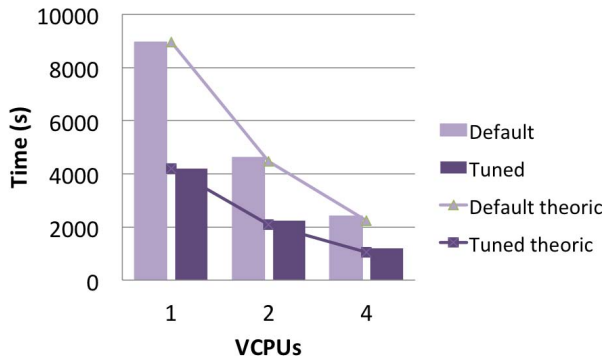

Fig. 5. Execution time for HandBrake video conversion.

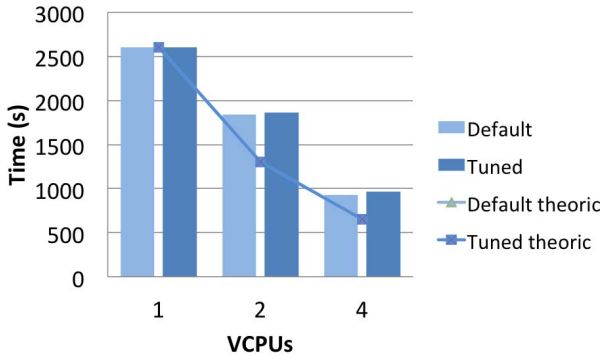

Fig. 6. Execution time for 3D DD-DG simulator.

The results of the Intel Linpack benchmark, employing 1, 2 and 4 VCPUs, are depicted in Fig. 4. The columns represent the performance in GFLOPS obtained for the default Apache CloudStack agent and also the tuned one. As we can see, the performance of the Apache CloudStack tuned agent is in average a 13.1% greater than the default one.

The results for the HandBrake case are depicted in Fig. 5. This figure represents the elapsed time in seconds necessary to perform the video conversion from MTS to mp4 format for both the default Apache CloudStack agent and the tuned one. As we can see, the time necessary to convert the video using the Apache CloudStack tuned agent is in average 51,7% lower than the corresponding time for the default agent. The reason of this improvement is that HandBrake can now take advantage of the extra processor features that are now available for the VM, such as SSSE3 and SSE4.2.

The results for the 3D DD-DG simulator are shown in Fig. 6. In this case, the columns represent, as in the previous example, the simulation time necessary to get the results for the Apache CloudStack tuned agent and also for the default agent. As we can see, there is no noticeable effect on the performance of the 3D DD-DG simulator in both Apache CloudStack agent implementations. This behavior is due to the fact that the 3D DD-DG simulator was not coded for taking advantage of the extra features exposed to the VM.

So, the performance obtained by the optimized version of the Apache CloudStack agent depends on the executed application. Applications that were not specifically coded to take advantage of the extra features of modern processors may not increase their performance. However, there are other applications that can take advantage of the extra processor features. If all the hypervisor-enabled hosts under a Pod have these features, the compatibility to perform the live migration of the VM can be retained.

V. Conclusions

The default KVM configuration employed by OpenNebula, Eucalyptus and Apache CloudStack provide a basic abstraction of the underlying CPU and this can produce a negative impact in the performance of the software executed in the VMs. For this reason, we have proposed an alternative to the default KVM configuration that enables the guest CPU to take advantage of extra instruction sets that are present in modern processors. The proposed optimization was implemented in the Apache CloudStack platform but could be implemented in other cloud platforms.

The tuned agent was tested in a cloud infrastructure based on Apache CloudStack 4.0.1, employing three applications: the Intel Linpack benchmark, the video transcoder HandBrake, and the 3D DD-DG simulator, an in-house parallel three-dimensional drift-diffusion semiconductor device simulator. In this case, the application was not coded to take advantage of the extra features and we have not found an increase in the performance. However, the results show that some applications such as the Intel Linpack benchmark and Handbrake exhibit increased performance. In the Intel Linpack case, the gain in the performance is slightly greater than 10% whereas in the case of HandBrake, the performance obtained employing the Apache CloudStack tuned agent is the double of the performance obtained with the default configuration.

Acknowledgment

This work has been supported by FEDER funds, by Spanish Government (MCYT) under project TEC2010-17320 and by Marie-Curie Fellowship under Grant PIEF-GA-2011-299990.

References

[1] B. Chee and C. Franklin, *Cloud Computing. Technologies and Strategies of the Ubiquitous Data Center*. Boca Raton: CRC Press, 2010.

[2] J. Rittinghouse and J. Ransome, *Cloud Computing: Implementation, Management, and Security*. Boca Raton: CRC Press, 2010.

[3] A. Velte, T. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*. USA: McGrawHill, 2010.

[4] KVM. http://www.linux-kvm.org

[5] R. Moreno-Vozmediano and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, Dec. 2012.

[6] Eucalyptus. http://open.eucalyptus.com

[7] Apache CloudStack. http://cloudstack.apache.org

[8] The Virtualization API. http://libvirt.org

[9] Intel Linpack. http://software.intel.com

[10] HandBrake. http://handbrake.fr

[11] N. Seoane, G. Indalecio, E. Comesana, M. Aldegunde, A. Garcia-Loureiro, and K. Kalna, "Random Dopant, Line-Edge Roughness, and Gate Workfunction Variability in a Nano InGaAs FinFET," *IEEE Trans. Electron Devices*, vol. 61, no. 2, pp. 466 –472, Feb. 2014.

[12] UNAH TV. http://www.unah.tv