Cost and Performance Analysis of MPI-Based SaaS on the Private Cloud Infrastructure

Oleg Bystrov(✉), Arnas Kačeniauskas, and Ruslan Pacevič

Vilnius Gediminas Technical University, 10223 Vilnius, Lithuania
oleg.bystrov@vilniustech.lt

Abstract. The paper presents the cost and performance analysis of parallel MPI-based software as a service (SaaS) deployed on the OpenStack cloud infrastructure. The parallel SaaS was developed by using C++ programming language and MPI library for the scientific discrete element method (DEM) computations of granular flows. The performance measured on KVM-based virtual machines was slightly higher than that on Docker containers of the OpenStack cloud. Round up and proportional pricing schemes were examined and compared from the user's perspective. The difference in cost computed by using alternative pricing schemes varied from 0.6% to 15.4%. However, this difference can be reduced to 1.0%, increasing execution time of considered tasks. The investigation of a trade-off between the execution time and cost was performed by using Pareto front analysis and a linear scalarization method. Bi-objective decision making revealed the preferable configurations of virtual machines specific to memory bound DEM computations, exploiting higher bandwidth.

Keywords: Cost and Performance Trade-off · Pareto Front · MPI · OpenStack

1 Introduction

In recent years, cloud computing has gained great popularity and transformed the IT industry [1]. Cloud computing infrastructures can provide the scalable resources on-demand to deploy performance and cost effective services. The NIST SPI model [2] represents a layered, high-level abstraction of cloud services classified into three main categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Organizations can use different implementations of cloud software for deploying their own private clouds. OpenStack [3] is an open source cloud management platform that delivers an integrated foundation to create, deploy and scale a secure and reliable public or private cloud. Another open source local cloud framework is Eucalyptus [4], provided by Eucalyptus Systems, Inc.

Cloud computing makes extensive use of virtual machines (VMs) because they allow workloads to be isolated and resource usage to be controlled. Kernel

Virtual Machine (KVM) [5] is a feature of Linux that allows Linux to act as a type 1 hypervisor, running an unmodified guest operating system inside a Linux process. Containers present an emerging technology for improving the productivity and code portability in cloud infrastructures. Due to the layered file system, Docker [6] container images require less disk space and I/O than the equivalent VM disk images. Thus, Docker has emerged as a standard runtime, image format and build system for Linux containers. IBM has added Docker container integration to Platform LSF to run the containers on an HPC cluster [7]. EDEM software has been deployed on Rescale's cloud simulation platform for high-performance computations [8]. However, it is difficult to provide precise guidelines regarding the optimal cloud platform and virtualization technology for each type of research and application [9].

Deployment of scientific codes as software services for data preparation, high-performance computation and visualization on the cloud infrastructure increases the mobility of users and achieves better exploitation. Thus, flexible cloud infrastructures and software services are perceived as a promising avenue for future advances in the multidisciplinary area of discrete element method (DEM) applications [8]. However, the cloud SaaS might suffer from severe performance degradation due to higher latencies of networks, virtualization overheads and other issues [1]. Cloud computing still lacks cost and performance analyses in the case of specific MPI-based applications, such as granular materials. Most evaluations of the virtualization overhead and performance of cloud services are based on standard benchmarks or theoretical unrealistic load models [9], therefore, the impact of the cloud infrastructure on the performance and cost of parallel MPI-based DEM computations remains unclear. Moreover, cost and performance are critical factors in deciding whether cloud infrastructures are viable for scientific DEM software.

The performance of virtual machines and lightweight containers has already received some attention in the academic literature [10–13]. However, few studies include the performance analysis of the virtualized distributed memory architectures for parallel MPI-based applications [14–16]. Bag-of-gangs applications [17] consist of parallel jobs that are in very frequent communication and must execute simultaneously and concurrently. Moschakis and Karatza [18] evaluated gang scheduling performance in the Amazon EC2 cloud. Sood [19] compared gang scheduling algorithms to other scheduling mechanisms in cloud computing. Hao et al. [20] proposed a 0–1 integer programming for the gang scheduling. Their proposed method tried its best finishing more jobs and minimizing the average waiting time. Bystrov et al. [21] investigated a trade-off between the computing speed and the consumed energy of a real-life hemodynamic application on a heterogeneous cloud. Beloglazov et al. [22] have proposed a modified best-fit algorithm for energy-aware resource provisioning in data centers while continuing to deliver the negotiated service level agreement. The survey [23] concludes that there exists no predictive model today truly and comprehensively capturing performance and energy consumption of the highly heterogeneous and hierarchical architecture of the modern HPC node. Moreover, the cost analysis of the MPI-based computations was not performed in the above overviewed research.

The resource allocation problem in cloud computing has received a lot of attention mainly in terms of cost optimization. Malawski et al. [24] presented a model, which assumed multiple cloud providers offering computational and storage services. The considered optimization objective was to reduce the total cost under deadline constraints. Liu et al. [25] focused on cost minimization and guarantee of performance, proposing the least cost per connection algorithm, which chose the most cost-effective VMs from the available public clouds. Zhou et al. [26] developed two evolutionary algorithms to optimize cost and execution time of scheduling workflows. Genez et al. [27] proposed an integer linear programming-based VM scheduler to produce low-cost scheduling for workflows execution in multiple cloud providers. Entrialgo et al. [28] designed a state-of-the-art cost optimization tool for the optimal allocation of VMs in hybrid clouds. Rosa et al. [29] developed the computational resource and cost prediction service, which measured user resources and reported the runtime financial cost before starting the workflow execution. A comprehensive review of workload scheduling and resource provisioning in cloud environments can be found in Wang et al. [30]. The most authors considered the total cost as the objective and solved the optimization problem with deadline constraint, which did not minimize the execution time, reducing its importance. Moreover, parallel MPI-based scientific applications were rarely examined because of their intensive communications between VMs and complex non-monotonous performance profiles.

The remaining paper is organized as follows: Sect. 2 outlines the governing relations of the discrete element method, Sect. 3 describes parallel MPI-based SaaS deployed on the OpenStack cloud infrastructure, Sect. 4 presents the cost and performance analysis and the conclusions are given in Sect. 5.

2 The Governing Relations of the Discrete Element Method

The discrete element method is a class of numerical techniques to simulate granular materials [31]. The frictional visco-elastic particle system consists of the finite number of deformable spherical particles with the specified size distribution and material properties. Any particle i in the system of N spherical particles undergoes the translational and rotational motion, involving the forces and torques originated in the process of their interaction. Finally, the motion of the i-th contacting spherical particle in time t is described as follows:

$$m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} = \boldsymbol{F}_i, I_i \frac{d\boldsymbol{\omega}_i}{dt} = \boldsymbol{T}_i, \tag{1}$$

where m_i and I_i are the mass and the moment of inertia of the particle, respectively, while the vectors $\boldsymbol{x}_i$ and $\boldsymbol{\omega}_i$ initiate the position of the centre of particle i and the rotational velocity around the particle centre of mass. The vectors $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$ present the resultant force and the resultant torque, acting in the centre of the particle i. The vector $\boldsymbol{F}_i$ can be expressed by the external force and the sum of the contact forces between the interacting particles:

$$\boldsymbol{F}_i = \boldsymbol{F}_{i,cont} + \boldsymbol{F}_{i,ext} = \sum_{j=1, j\neq i}^{N} \boldsymbol{F}_{ij,cont} + m_i, \boldsymbol{g}, \tag{2}$$

where $\boldsymbol{F}_{i,ext}$ and $\boldsymbol{F}_{i,cont}$ are the external force and the resultant contact force of particle i, respectively, $\boldsymbol{g}$ is the acceleration due to gravity, $\boldsymbol{F}_{ij,cont}$ is the interparticle contact force vector, describing the contact between the particles i and j. Thus, in the present work, the electromagnetic force [32], the aerodynamic force [33] and other external forces [34,35], except for the gravity force are not considered. The rotational motion is governed by particle torques $\boldsymbol{T}_i$ that can be expressed by torques $\boldsymbol{T}_{ij}$ of the neighbouring particles:

$$\boldsymbol{T}_i = \sum_{j=1, j\neq i}^{N} \boldsymbol{T}_{ij} = \sum_{j=1, j\neq i}^{N} \boldsymbol{d}_{cij} \times \boldsymbol{F}_{i,cont}, \tag{3}$$

where $\boldsymbol{d}_{cij}$ is the vector pointing from the particle centre to the contact centre. The interparticle contact force vector $\boldsymbol{F}_{i,cont}$ may be expressed in terms of normal and tangential components. The normal component of the contact force comprises the elastic counterpart according to Hertz theory and the viscous counterpart that can be represented by the spring-dashpot model [36] as follows:

$$\boldsymbol{F}_{ij,n} = \frac{4}{3} \cdot \frac{E_i E_j}{E_i(1-\nu_j^2) + E_j(1-\nu_i^2)} R_{ij}^{1/2} \delta_{ij,n}^{3/2} \boldsymbol{n}_{ij} - \gamma_n m_{ij} \boldsymbol{v}_{ij,n}, \tag{4}$$

where $\boldsymbol{n}_{ij}$ is the normal vector, R_{ij} is the reduced radius of the contacting particles, γ_n is the constant normal damping coefficient, m_{ij} is the reduced mass of the contacting particles and $\boldsymbol{v}_{ij,n}$ is the normal component of the relative velocity of the contact point. E_i and E_j are elastic moduli, ν_i and ν_j are Poison's ratios of contacting particles i and j, respectively. In the normal direction, the depth of the overlap between particles i and j is defined by $\delta_{ij,n}$.

The evolution of the tangential contact force can be divided into the parts of static friction prior to sliding $\boldsymbol{F}_{ij,stat,t}$ and dynamic slip friction $\boldsymbol{F}_{ij,dyn,t}$ [36]:

$$\boldsymbol{F}_{ij,t} = -\boldsymbol{t}_{ij} \begin{cases} |\boldsymbol{F}_{ij,stat,t}|, & |\boldsymbol{F}_{ij,stat,t}| < \mu |\boldsymbol{F}_{ij,n}| \\ |\boldsymbol{F}_{ij,dyn,t}|, & |\boldsymbol{F}_{ij,stat,t}| \geq \mu |\boldsymbol{F}_{ij,n}| \end{cases}, \tag{5}$$

where $\boldsymbol{t}_{ij}$ is the unit vector of the tangential contact direction. The model of static friction force is implemented, when the tangential force is smaller than the Coulomb-type cut-off limit. In the opposite case, the dynamic friction expressed by the normal contact force and the Coulomb friction coefficient μ is considered:

$$\boldsymbol{F}_{ij,dyn,t} = -\mu |\boldsymbol{F}_{ij,n}| \boldsymbol{t}_{ij}, \tag{6}$$

The static friction force is calculated by summing up the elastic and viscous damping components [37]:

$$\boldsymbol{F}_{ij,stat,t} = -\frac{16}{3} \cdot \frac{G_i G_j \sqrt{R_{ij}\delta_{ij,n}}}{G_i(2-\nu_j) + G_j(2-\nu_i)} |\delta_{ij,t}| \boldsymbol{t}_{ij} - \gamma_t m_{ij} \boldsymbol{v}_{ij,t}, \tag{7}$$

where $|\delta_{ij,t}|$ is the length of tangential displacement, $\boldsymbol{v}_{ij,t}$ is the tangential component of the relative velocity of the contact point, γ_t is the constant tangential damping coefficient, while G_i and G_j are shear moduli of the particles i and j, respectively.

The main CPU-time-consuming computational procedures of the DEM are contact detection, contact force computation and time integration. Contact detection was based on the simple and fast implementation of a cell-based algorithm [38]. The explicit velocity Verlet algorithm [38] was used for time integration employing small time steps. The details of outlined DEM model (1–7) and its implementation can be found in [36,39].

3 DEM SaaS Deployed on OpenStack Cloud

The parallel DEM software was developed and deployed as SaaS on the cloud infrastructure to perform time-consuming computations of granular materials.

3.1 Parallel DEM SaaS

The simulation of systems at the particle level of detail has the disadvantage of making DEM computationally very expensive. The selection of an efficient parallel solution algorithm depends on the specific characteristics of the considered problem and the numerical method used [39–41]. The parallel DEM algorithms differ from the analogous parallel processing in the continuum approach. Moving particles dynamically change the workload configuration, making parallelization of DEM software much more difficult and challenging. Domain decomposition is considered one of the most efficient coarse grain strategies for scientific and engineering computations, therefore, it was implemented in the developed DEM code. The recursive coordinate bisection (RCB) method from the Zoltan library [42] was used for domain partitioning because it is highly effective for particle simulations. The RCB method recursively divides the computational domain into nearly equal subdomains by cutting planes orthogonal to the coordinate axes, according to particle coordinates and workload weights. This method is attractive as a dynamic load-balancing algorithm because it implicitly produces incremental partitions and reduces data transfer between processors caused by repartitioning.

The employed DEM software was developed using C++ programming language. Interprocessor communication was implemented in the DEM code by subroutines of the message passing library MPI. Each processor computes the forces and updates the positions of particles only in its subdomain. To perform their computations, the processors need to share information about particles that are near the division boundaries in ghost layers. The main portion of communications is performed prior to performing contact detection and contact force computation. In the present implementation, particle data from the ghost layers are exchanged between neighboring subdomains. The exchange of positions and velocities of particles between MPI processes is a common strategy often used in DEM codes [43]. Despite its local character, interprocessor particle data transfer requires a significant amount of time and reduces the parallel efficiency of computations. The parallel DEM software was deployed on the cloud infrastructure by developing the environment launchers designed for users to configure the SaaS and define custom settings. After successful authorization, the user can define configuration parameters and run the parallel SaaS on ordered virtual resources.

3.2 OpenStack Cloud Infrastructure

The university private cloud infrastructure based on OpenStack Train 2019 version [3] is hosted in the Vilnius Gediminas Technical University. The deployed capabilities of the OpenStack cloud infrastructure include compute service Nova, compute service Zun for containers, networking service Neutron, container network plugin Kuryr, image service Glance, identity service Keystone, object storage service Swift and block storage service Cinder. Nova automatically deploys the provisioned virtual compute instances (VMs), Zun launches and manages containers, Swift provides redundant storage of static objects, Neutron manages virtual network resources, Kuryr connects containers to Neutron, Keystone is responsible for authentication and authorization, while Glance provides service discovery, registration and delivery for virtual disk images.

The cloud infrastructure is managed by the OpenStack API, which provides access to infrastructure services. The OpenStack cloud IaaS provides platforms (PaaS) to develop and deploy software services called SaaS. The PaaS layer supplies engineering application developers with programming-language-level environments and compilers, such as GNU compiler collection. Parallel software for distributed memory systems is developed using the Open MPI platform, which includes the open source implementation of the MPI standard for message passing. The development platform as a service for domain decomposition and dynamic load balancing is provided based on the Zoltan library [42]. It simplifies the load-balancing and data movement difficulties that arise in dynamic simulations. The DEM SaaS was deployed on top of the provided platforms, such as GNU compiler collection, the message passing library Open MPI and the Zoltan library. Computational results are visualized using the cloud visualization service VisLT [44].

The cloud infrastructure is composed of OpenStack service nodes and compute nodes (Intel®Core i7-6700 3.40 GHz CPU, 32 GB DDR4 2133 MHz MHz RAM and 1 TB HDD) connected to 1 Gbps Ethernet LAN. Two alternatives of the virtualization layer are implemented to gain more flexibility and efficiency in resource configuration. Version 2.11.1 of QEMU-KVM is used for virtual machines (VMs) deployed and managed by Nova. Alternatively, Docker version

Table 1. Characteristics of virtual machines and containers.

	Cores	CPU type	RAM, GB	HDD, TB	Price, $/h
VM.small	1	i7-6700	8	0.5	0.0455
VM.small	1	i7-6700	8	0.5	0.0455
VM.small	1	i7-6700	8	0.5	0.0455

19.03.6 containers (CNs) launched and managed by Zun create an abstraction layer between computing resources and the services using them. Ubuntu 18.04 LTS (Bionic Beaver) is installed in the VMs and CNs. Characteristics and prices of VMs and CNs are provided in Table 1. Monetary costs of allocated VMs/CNs are described by price per hour according to Amazon EC2 VM type C5. Two pay-per-use pricing schemes are considered for all VM/CN types. In the case of the traditional cloud pricing scheme named round up, VM instances are billed per hour of usage, but each partial instance-hour is billed as a full hour. In the case of the pricing scheme named proportional, the cost is directly proportional to the time the VMs are allocated, which corresponds to price per second scheme.

4 The Cost and Performance Analysis

The cost and performance of the developed DEM SaaS for parallel computations of granular flows is investigated. The gravity packing problem of granular material, falling under the influence of gravity into a container, was considered because it often served as a benchmark for performance measurements [16]. The solution domain was assumed to be a cubic container with the 1.0m-long edges.

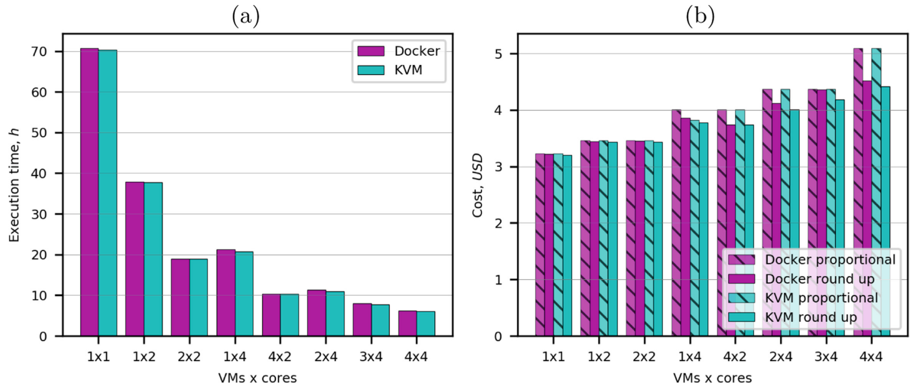

Fig. 1. Execution time and cost: (a) the execution time on KVM virtual machines and Docker containers, (b) the cost computed by using round up and proportional pricing schemes.

Half of the domain was filled with 1000188 monosized particles, using a cubic structure. Performing the benchmark on VMs and CNs of OpenStack cloud, the computation time of 200000 time steps equal to 1.0x10^{-6} was measured.

Figure 1 presents the SaaS execution time and cost on KVM VMs and Docker CNs measured for different numbers of VMs/CNs and cores used. Higher computational performance of DEM SaaS was observed on KVM virtual machines, but the measured difference did not exceed 3.9% of execution time on Docker CNs. Speedup of parallel computations equal to 11.6 was measured on 4x4 configuration of KVM VMs (16 cores), which gave parallel efficiency equal to 0.73. The measured speedup values are close to those obtained for relevant numbers of cores in other parallel performance studies of DEM software [16,43]. The obvious difference in cost computed by using alternative pricing schemes can be observed. This difference varied from 0.6% to 15.4%, depending on the number of VMs or CNs used.

Figure 2 shows the relative difference in cost calculated by using two pricing schemes for various software execution times. The execution time of the numerical DEM software almost linearly depends on the number of time steps used for time integration of Eq. (1). Thus, the number of computed time steps provides the length of the simulated physical time interval, which represents the amount of computations. It can be observed that the difference decreased when longer tasks were executed. The difference diminished to 1.0% in the case of 1600000 computed time steps. Moreover, larger differences caused by multi-node and multi-core execution of MPI-based SaaS can be observed for larger number of VMs/CNs and cores in spite of scattered results.

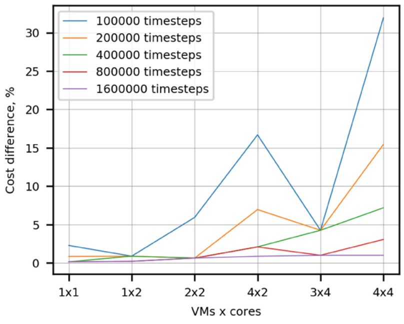

Fig. 2. The relative difference in cost calculated by using round up and proportional pricing schemes for various execution times on KVM virtual machines.

The choice of the optimal hardware setup needs to be taken in the presence of two conflicting objectives or criteria: the execution time T and the computation cost C. This bi-objective optimization problem can be formulated as follows:

$$\min_{p_i \in X} (T(p_i), C(p_i)), \tag{8}$$

where $X = \{1\text{x}1,\ 1\text{x}2,\ 2\text{x}2,\ 1\text{x}4,\ 4\text{x}2,\ 2\text{x}4,\ 3\text{x}4,\ 4\text{x}4\}$ is the set of feasible solutions. The alternative VMs/CNs configurations 2x2 and 1x4 mean 2 VM.medium instances with 2 cores on 2 nodes and 1 VM.large instance with 4 cores on 1 node, respectively.

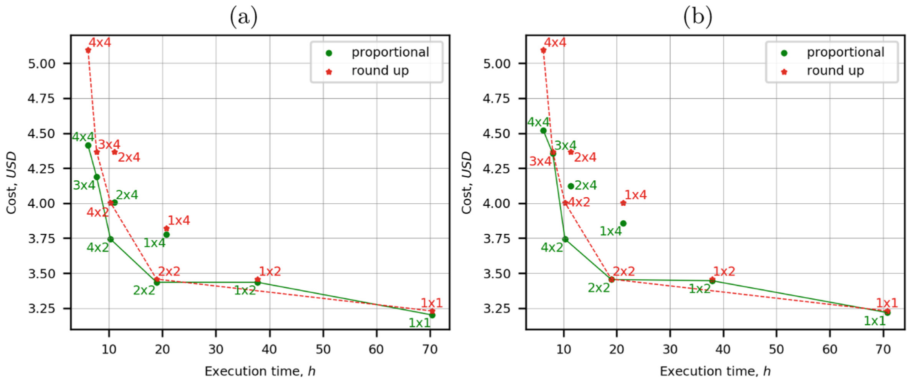

Fig. 3. Pareto fronts for alternative pricing schemes: (a) KVM VMs, (b) Docker CNs.

There are many different approaches to deal with multi-objective optimization problems. A common approach is to find the Pareto optimal solutions, i.e., the solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives. For the formulated bi-objective optimization problem (8), the Pareto optimal solutions are presented in Fig. 3. It was expected that the proportional pricing scheme dominated over the round up pricing scheme and was preferable for users. Solutions based on KVM VMs were better than that based on Docker CNs, but the difference was not large in most cases. The VMs configuration 1x2 belonged to Pareto front in the case of the proportional pricing scheme, but it was excluded from the Pareto front in the case of the round up pricing scheme. It is worth noting that the VMs configurations 2x2 and 4x2 were always preferable over 1x4 and 2x4, which was specific to memory bound DEM computations exploiting higher bandwidth.

Scalarization is as a popular approach to solve a multi-objective optimization problem, considering subjective preferences of a decision maker. The original problem (8) is converted to a single-objective optimization problem by using user defined weights w_T and w_C for normalized execution time objective and normalized cost objective, respectively. Figure 4 shows dependency of scalarized objective function on VMs/CNs configuration for equal (Fig. 4a) and execution time oriented (Fig. 4b) weights. The difference between pricing schemes can be clearly observed only for VMs/CNs configurations with the total number of cores larger than 4. The equal weights resulted in optimal VMs/CNs configuration 2x2, while execution time-oriented weights gave the optimal configuration 4x2. DEM SaaS computations on VMs/CNs configurations 2x2 and 4x2 were so fast

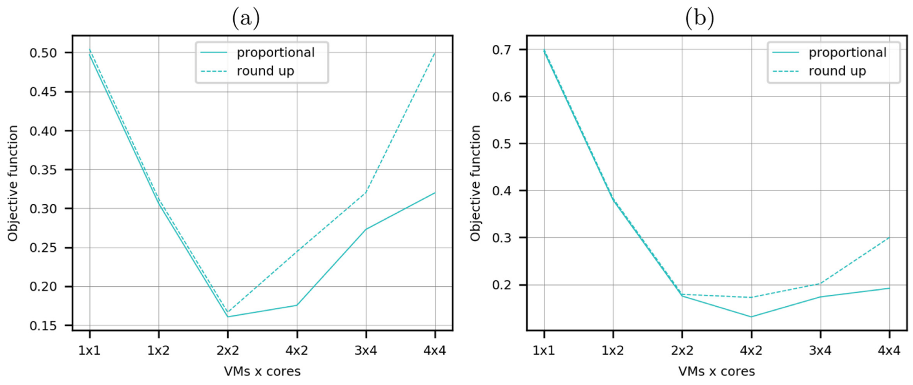

Fig. 4. The application of linear scalarization method: (a) the equal weights ($w_T = 0.5$ and $w_C = 0.5$), (b) the execution time oriented weights ($w_T = 0.7$ and $w_C = 0.3$).

(Fig. 1a) that they dominated over other solutions in the wide range of weights values.

5 Conclusions

In this article, cost and performance analysis of MPI-based computations performed by the discrete element method SaaS on KVM virtual machines and Docker containers of the OpenStack cloud is presented. The SaaS execution time measured on KVM virtual machines was shorter by 0.3–3.9% than that on Docker containers. The difference in cost computed by using alternative pricing schemes varied from 0.6% to 15.4%, depending on the number of virtual machines or containers used. However, the difference decreased to 1.0% for 8 times longer tasks. Pareto front and linear scalarization revealed the preferable VMs/CNs configurations specific to memory bound DEM computations exploiting higher bandwidth.

References

1. Khan, A.A., Zakarya, M.: Energy, performance and cost efficient cloud datacentres: a survey. Comput. Sci. Rev. **40**, 100390 (2021)
2. Mell, P.M., Grance, T.: The NIST definition of cloud computing. Technical report (2011)
3. Openstack. https://www.openstack.org. Accessed 9 Apr 2022
4. Nurmi, D., et al.: The Eucalyptus open-source cloud-computing system. In: 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131. IEEE (2009)
5. Chierici, A., Veraldi, R.: A quantitative comparison between XEN and KVM. J. Phys: Conf. Ser. **219**(4), 1–10 (2010)
6. Docker. https://www.docker.com. Accessed 9 Apr 2022

7. McMillan, B., Chen, C.: High performance docking. Technical report (2014)
8. Edem now available on rescale's cloud simulation platform. https://www.edemsimulation.com/blog-and-news/news/edem-now-available-rescales-cloud-simulation-platform/. Accessed 9 Apr 2022
9. Sakellari, G., Loukas, G.: A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. Simul. Model. Pract. Theory **39**, 92–103 (2013)
10. Kačeniauskas, A., et al.: Private cloud infrastructure for applications of mechanical and medical engineering. Inf. Technol. Control **44**(3), 254–261 (2015)
11. Kozhirbayev, Z., Sinnott, R.O.: A performance comparison of container-based technologies for the cloud. Futur. Gener. Comput. Syst. **68**, 175–182 (2017)
12. Chae, M., Lee, H., Lee, K.: A performance comparison of linux containers and virtual machines using docker and KVM. Clust. Comput. **22**(S1), 1765–1775 (2017)
13. Potdar, A.M., Narayan, G.D., Kengond, S., Mulla, M.M.: Performance evaluation of docker container and virtual machine. Procedia Comput. Sci. **171**, 1419–1428 (2020)
14. Hale, J.S., Li, L., Richardson, C.N., Wells, G.N.: Containers for portable, productive, and performant scientific computing. Comput. Sci. Eng. **19**(6), 40–50 (2017)
15. Mohammadi, M., Bazhirov, T.: Comparative benchmarking of cloud computing vendors with high performance Linpack. In: Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications - HP3C. ACM Press (2018)
16. Bystrov, O., Pacevič, R., Kačeniauskas, A.: Performance of communication- and computation-intensive SaaS on the OpenStack cloud. Appl. Sci. **11**(16), 7379 (2021)
17. Papazachos, Z.C., Karatza, H.D.: Performance evaluation of bag of gangs scheduling in a heterogeneous distributed system. J. Syst. Softw. **83**(8), 1346–1354 (2010)
18. Moschakis, I.A., Karatza, H.D.: Evaluation of gang scheduling performance and cost in a cloud computing system. J. Supercomput. **59**(2), 975–992 (2010)
19. Sood, K.: Comparative study of scheduling mechanisms in cloud computing. IOSR J. Eng. **4**(5), 30–33 (2014)
20. Hao, Y., Liu, G., Hou, R., Zhu, Y., Lu, J.: Performance analysis of gang scheduling in a grid. J. Netw. Syst. Manage. **23**(3), 650–672 (2014)
21. Bystrov, O., et al.: Performance evaluation of parallel haemodynamic computations on heterogeneous clouds. Comput. Inform. **39**(4), 695–723 (2020)
22. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur. Gener. Comput. Syst. **28**(5), 755–768 (2012)
23. O'Brien, K., Pietri, I., Reddy, R., Lastovetsky, A., Sakellariou, R.: A survey of power and energy predictive models in HPC systems and applications. ACM Comput. Surv. **50**(3), 1–38 (2017)
24. Malawski, M., Figiela, K., Nabrzyski, J.: Cost minimization for computational applications on hybrid cloud infrastructures. Futur. Gener. Comput. Syst. **29**(7), 1786–1794 (2013)
25. Luo, B., Niu, Y., Liu, F.: Cost-effective service provisioning for hybrid cloud applications. In: Guo, S., Liao, X., Liu, F., Zhu, Y. (eds.) CollaborateCom 2015. LNICST, vol. 163, pp. 47–56. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28910-6_5
26. Zhou, J., Wang, T., Cong, P., Lu, P., Wei, T., Chen, M.: Cost and makespan-aware workflow scheduling in hybrid clouds. J. Syst. Architect. **100**, 101631 (2019)

27. Genez, T.A., Bittencourt, L.F., Madeira, E.R.: Time-discretization for speeding-up scheduling of deadline-constrained workflows in clouds. Futur. Gener. Comput. Syst. **107**, 1116–1129 (2020)
28. Entrialgo, J., García, M., Díaz, J.L., García, J., García, D.F.: Modelling and simulation for cost optimization and performance analysis of transactional applications in hybrid clouds. Simul. Model. Pract. Theory **109**, 102311 (2021)
29. Rosa, M.J., Ralha, C.G., Holanda, M., Araujo, A.P.: Computational resource and cost prediction service for scientific workflows in federated clouds. Futur. Gener. Comput. Syst. **125**, 844–858 (2021)
30. Wang, B., Wang, C., Song, Y., Cao, J., Cui, X., Zhang, L.: A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds. Clust. Comput. **23**(4), 2809–2834 (2020). https://doi.org/10.1007/s10586-020-03048-8
31. Cundall, P.A., Strack, O.D.L.: A discrete numerical model for granular assemblies. Géotechnique **29**(1), 47–65 (1979)
32. Tumonis, L., Schneider, M., Kačianauskas, R., Kačeniauskas, A.: Comparison of dynamic behaviour of EMA-3 railgun under differently induced loadings. Mechanika **78**(4), 31–37 (2009)
33. Kačeniauskas, A., Rutschmann, P.: Parallel FEM software for CFD problems. Informatica **15**(3), 363–378 (2004)
34. Liu, G., Marshall, J.S., Li, S.Q., Yao, Q.: Discrete-element method for particle capture by a body in an electrostatic field. Int. J. Numer. Meth. Eng. **84**(13), 1589–1612 (2010)
35. Tumonis, L., Kačianauskas, R., Kačeniauskas, A., Schneider, M.: The transient behavior of rails used in electromagnetic railguns: numerical investigations at constant loading velocities. J. Vibroeng. **9**, 15–17 (2007)
36. Džiugys, A., Peters, B.: An approach to simulate the motion of spherical and non-spherical fuel particles in combustion chambers. Granul. Matter **3**(4), 231–266 (2001)
37. Kohring, G.A.: Studies of diffusional mixing in rotating drums via computer simulations. J. Phys. I **5**(12), 1551–1561 (1995)
38. Norouzi, H.R., Zarghami, R., Sotudeh-Gharebagh, R., Mostoufi, N.: Coupled CFD-DEM Modeling. Wiley, Chichester (2016)
39. Kačeniauskas, A., Kačianauskas, R., Maknickas, A., Markauskas, D.: Computation and visualization of discrete particle systems on gLite-based grid. Adv. Eng. Softw. **42**(5), 237–246 (2011)
40. Šešok, D., Belevičius, R., Kačeniauskas, A., Mockus, J.: Application of GRID computing for optimization of grillages. Mechanika **82**(2), 63–69 (2010)
41. Stupak, E., et al.: The geometric model-based patient-specific simulations of turbulent aortic valve flows. Arch. Mech. **69**(4–5), 317–345 (2017)
42. Devine, K., Boman, E., Heaphy, R., Hendrickson, B., Vaughan, C.: Zoltan data management services for parallel dynamic applications. Comput. Sci. Eng. **4**(2), 90–96 (2002)
43. Berger, R., Kloss, C., Kohlmeyer, A., Pirker, S.: Hybrid parallelization of the LIGGGHTS open-source DEM code. Powder Technol. **278**, 234–247 (2015)
44. Pacevič, R., Kačeniauskas, A.: The development of VisLT visualization service in Openstack cloud infrastructure. Adv. Eng. Softw. **103**, 46–56 (2017)