Toward a New Massively Distributed Virtual Machine based Cloud Micro-Services Team Model for HPC: SPMD Applications

Fatéma Zahra Benchara
Department of Computer Science
Laboratory SSDIA, ENSET Mohammedia, Hassan II
University of Casablanca
Mohammedia, Morocco

Omar Bouattane
Department of Computer Science
Laboratory SSDIA, ENSET Mohammedia, Hassan II
University of Casablanca
Mohammedia, Morocco

Mohamed Youssfi
Department of Computer Science
Laboratory SSDIA, ENSET Mohammedia, Hassan II
University of Casablanca
Mohammedia, Morocco

Ouafae Serrar
Department of Computer Science
CRMEF MARRAKECH-SAFI,
MARRAKECH, Morocco

Hassan Ouajji
Laboratory SSDIA, ENSET Mohammedia, Hassan II
University of Casablanca
Mohammedia, Morocco

***Abstract*—This paper aims to propose a new massively distributed virtual machine with scalable and efficient parallel computing models for High Performance Computing (HPC). The message passing paradigm of the Processing Units has a significant impact on HPC with high communication cost that penalizes the performance of these models. Accordingly, the proposed micro-services model allows the HPC applications to enhance the processing power with low communication cost. Thus, the model based Micro-services Virtual Processing Units (MsVPUs) cooperate using asynchronous communication mechanism through the Advanced Message Queuing Protocol (AMQP) protocol in order to maintain the scalability of the Single Program Multiple Data (SPMD) applications. Additionally, this mechanism enhances also the efficiency of the model based load balancing service with time optimized load balancing strategy. The proposed virtual machine is tested and validated through an application of fine grained parallel programs for big data classification. Experimental results present reduced execution time compared to the virtual machine based mobile agent's model.**

***Keywords*—Parallel and distributed computing; micro-services; cloud computing; distributed virtual machine; high performance computing**

I. Introduction

Recently, computer science application converges to HPC one. This is due to the new application expectations for Big data analysis [1], and real time information accessibility on multiple devices (Smartphones, Laptops, Tablets…). Thus, the data to be processed and the related complex computations oriented these applications to new HPC processing environments (clusters, grids and clouds [2], [3]) which provide the required processing power. The HPC systems based cloud computing are constituted by a set of distributed heterogeneous machines connected through an interconnection network and collaborate by their own resources in order to provide the processing power with an optimized computation time; such as in Amazon Elastic Compute Cloud (EC2) [4] that aims to enhance the execution of HPC applications in cloud. The collaboration between the distributed processing units is based on the HPC environment middleware which orchestrates the computation and manages the distribution of data and tasks between them. However, the performance of these environments is related to the one of their based middleware [5]. Normally, this middleware has to manage these two following major HPC challenges: 1) Message passing challenge the intensive communication between the computing units, has a great impact on the global computation time and the scalability of these applications, with the corresponding high communication cost. 2) Heterogeneity of computing nodes challenge the difference of nodes performance influence also the global computation time with an unbalanced computing environment caused by the overloaded workload of the slowest node. Indeed, the middleware based massively distributed computing environment has to deal with the above challenges in order to provide a scalable and efficient massively distributed computing environment. Thus, what are the promising paradigms for managing these challenges? This paper presents a new massively distributed virtual machine model based on cloud micro-services which aims to implement an asynchronous communication mechanism for computing units message passing. The main contributions of this paper are that 1) the proposed virtual machine considers providing the processing power needed for HPC applications by its integrated micro-services team model which is constituted by virtual computing units, and 2) considers the communication challenge using a lightweight communication mechanism, and also 3) considers the heterogeneity challenge by implementing a load balancing strategies. The paper is organized as follows:

- We present the virtual machine based cloud distributed computing model and its innovative components (Section 3) which are the micro-services.
- We demonstrate that the model based middleware is promising (Section 4); and that by implementing some SPMD applications (Section 5) we ensure a scalable and efficient cooperative parallel and distributed computing environment.

II. Background

To highlight the aim of this paper, we present the parallel and distributed computing [6] field and its key techniques for performing intensive computation in a few time. For example, in order to perform a password encryption program on 1000 passwords (Fig. 1) there are two main case study: 1) Sequential case where the program is performed on a single machine with an execution time TE per password, and the global computation time $Tt_{Seq}=\sum TE$. Despite, in 2) Parallel and Distributed computing case, the program is encapsulated on 10 machines which cooperate and distribute the data between them and work in parallel so that the global computation time will be reduced significantly with Tt(p&d)<< TtSeq. The last case will perform a high performance computing if the computing model integrates some mechanisms for parallel and distributed computing challenges; the communication and the load balancing challenges. So, the scalable computing model will be the one which can optimize significantly the global computation time. This model is implemented on parallel and distributed virtual machine that orchestrates and manages the distribution of data and tasks between the nodes.

There are several inspiring proposed parallel and distributed virtual machines [7]-[11] that used different technologies such as the MCC(Mesh Connected Computer) mesh and the FPGA (Field-Programmable Gate Array). However, the scalability and efficiency of these virtual machines depends on the ability of their corresponding middleware to handle the HPC computing challenges. The Middleware is the main components in the distributed systems that can manage a set of heterogeneous nodes. The Multi Agent System MAS is a promising technology for implementing such middleware. However, the micro-services implements the flexibility with the others technologies trends and the easy integration in cloud to improve HPC.

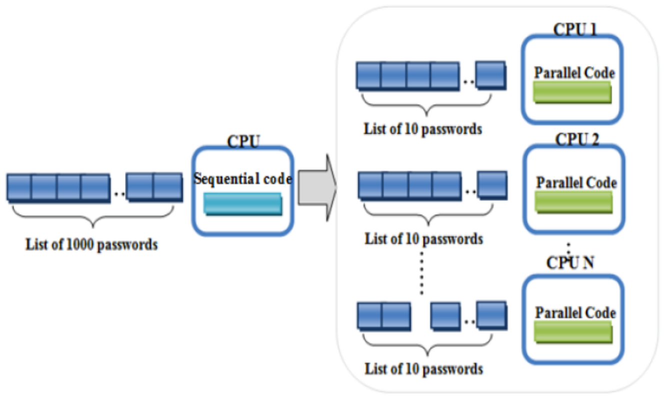


Fig. 1. Parallel and distributed computing paradigm.

III. Proposed Massively Distributed Virtual Machine

A. Massively Distributed Virtual Machine Architecture

The proposed massively distributed virtual machine is a new parallel and distributed computing environment, constituted over distributed heterogeneous nodes in distributed system. This virtual machine based micro-services model which is managed by cloud middleware, allows performing the parallel and distributed programs as services by cooperative micro-services team MsVPUs. For each deployed service, the Scientifics and researchers can take benefits of the flexibility of this virtual machine with the parallel computing models such as: SPMD, MPMD, and topologies (2D Mesh, 3D Mesh,…). Each MsVPU is an autonomous service that collaborates with the computing team using well determined communication mechanism for HPC. For example (Fig. 2), in order to perform the big data classification the well-known classification algorithms; c-means and Fuzzy c-means are implemented in this virtual machine as distributed classification service (Section 3) according to SPMD architecture. To do so, each team worker MsVPU will receive the input data from its team leader MsVPU, and perform the classification service and send the results back to its team leader in order to accomplish the execution of the application.

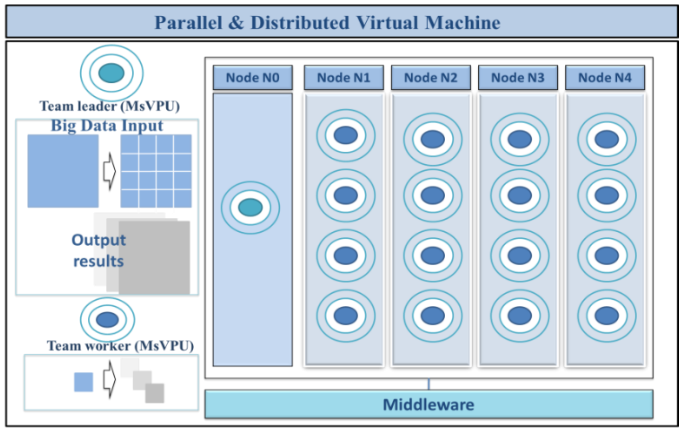


Fig. 2. SPMD distributed computing model based micro-services approach.

B. Distributed Computing Model based Main Components

In order to perform a high performance parallel and distributed computing, the proposed virtual machine model collaborate specific types of micro-services according to their tasks. When the parallel and distributed program is deployed on this virtual machine the micro-services team is created. This later is constituted by; Team leader (MsVPU) micro-service and the team workers (MsVPUs) that are distributed on each node to perform their corresponding services. Each team worker (MsVPU) encapsulates the program as service and collaborates with the other MsVPUs and provides the results to their Team leader (MsVPU) micro-service which manages and orchestrates the computing of its team while the execution of the program. This virtual machine allows deploying more than one parallel and distributed program by its integrated Proxy Ms Provider micro-service which works with the Load Balancer Ms micro-service in order to choose the appropriate team for each application request. The main principal micro-services of the model (Fig. 3) are presented as follows:

- **Proxy Ms Provider.** This micro-service is the mediator between the micro-services MsVPUs and the applications. The application requests are sent to this micro-service which communicate with the Load Balancer Ms in order to choose and send the request to the appropriate micro-service MsVPU. Then, the Proxy Ms Provider returns the results to the appropriate application.
- **Load Balancer Ms.** This micro-service is the one responsible of the management of the micro-services of the virtual machine. Each micro-service publishes its information (name, address, port, and number of CPUs) in this micro-service. So, this helps the Load Balancer Ms to get the node performance and ensure the load balancing of micro-services according to well defined load balancing strategies**.**
- **Team leader MsVPU.** This micro-service is the one responsible of the execution of the application requests. It cooperate with its team works (MsVPUs) in order to execute the parallel and distributed programs as services and sends the final results to the Proxy Ms Provider. This micro-service can be deployed in many distributed nodes.
- **Team worker MsVPUs.** This micro-service corresponds to a CPU. Each MsVPU receives the data from its team leader Ms and executes the service and returns the results to this later in order to compute the finale results.
- **DF Ms.** This micro-service centralizes the configuration of micro-services of the model. Each deployed micro-service will search for its configuration on this micro-service. So, the Proxy Ms Provider will easily follow the appropriate micro-services of the application request.

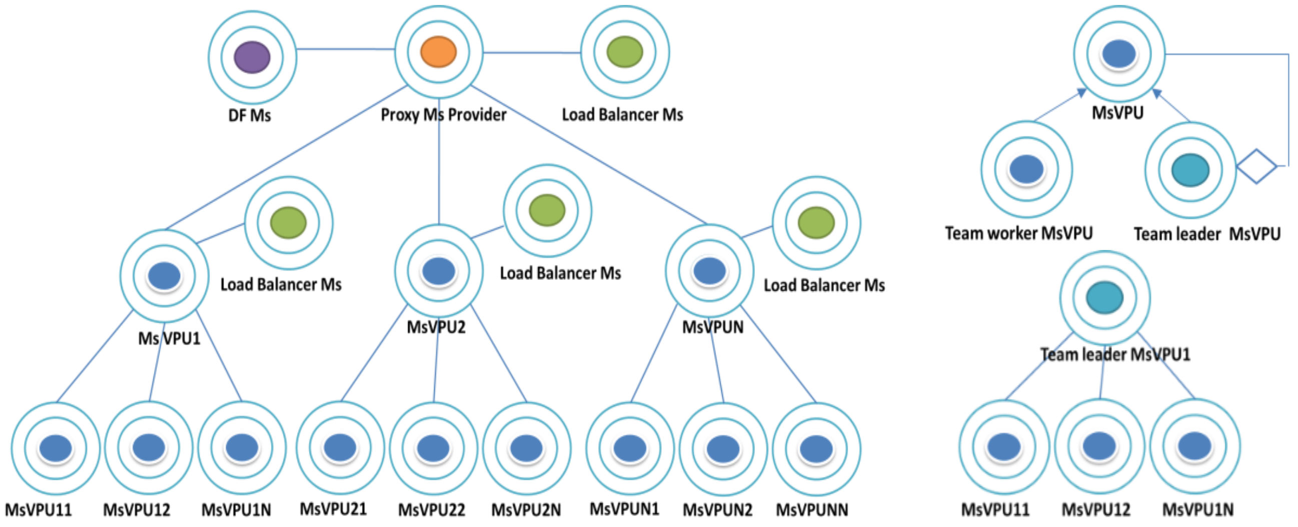


Fig. 3. Architecture of the Main components of the massively distributed virtual machine.

The UML diagram of the virtual machine model is illustrated in Fig. 4, which allows the MsVPUs micro-services to collaborate in the grid computing in order to perform the distributed services according to different programming models and parallel topologies.

The communication between the computing model main components is presented in the sequence diagram of Fig. 5. For example, in order to perform the parallel and distributed computing service, the application sends the request with the input data to Ms Proxy Provider. This later sends this request to the Ms Load Balancer which determines the Team leader Ms that will perform this request, and sends its address to the Ms Proxy Provider which sends the input data to the right Team leader Ms in order to perform this request in collaboration with its team of MsVPUs. At the end, the final result is send back to the application by the Ms Proxy provider.

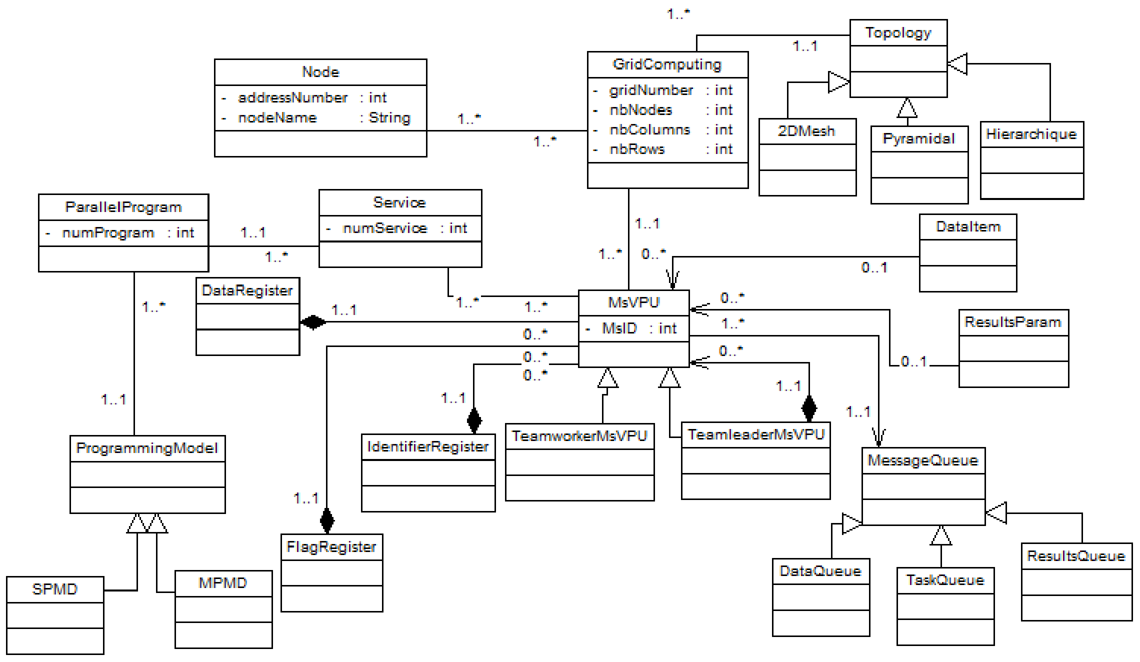


Fig. 4. UML diagram of the proposed massively distributed virtual machine model.

C. *Massively Distributed Computing Middleware*

The Massively Distributed computing Middleware (Fig. 6) is a new paradigm based micro-services, which allows dividing the complex tasks of the parallel programs to independent sub tasks as distributed micro-services deployed on the computing model of virtual machine. This computing model cooperates the micro-service team leader MsVPU and its micro-services team workers MsVPU in order to perform the parallel programs on cloud computing platform. So, the scalability and efficiency of this middleware are illustrated by its two main modules; Communication Optimization Module for implementing the asynchronous communication mechanism and Load Balancing Module in order to manage the overloads between the micro-services.

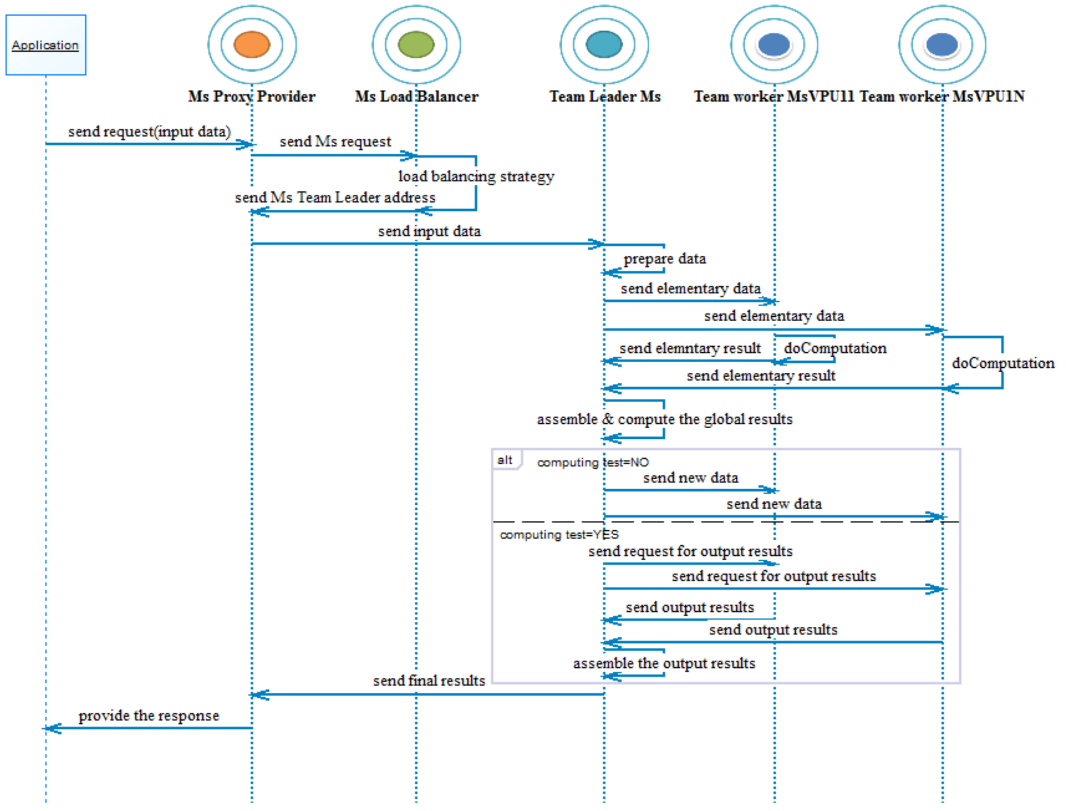


Fig. 5. Communication diagram of main components of the massively distributed virtual machine.

Communication Optimization Module This module ensures a lightweight communication mechanism between the micro-services of the computing model. This is done, by the implementation of the RabbitMQ messaging Framework in the computing model. So that the micro-services will use asynchronous communication by exchanging messages based on AMQP (Advanced Message Queuring Protocol) protocol. Furthermore, this module provides three types of message queues (data_queue, tasks_queue, and results_queue) which store and provide the exchanged messages between the micro-services. For example (Fig. 7) in order to perform an SPMD service, the Team leader Ms micro-service sends the computing data to the data_queue, and then this data is sent to the appropriate MsVPU micro-services. Each MsVPU will execute the service and send the results to the results_queue in order to be received by the Team leader Ms.

LoadBalancing Module This module provides a load balancing mechanism for the micro-services of the computing model by a specific micro-service the *Load Balancer Ms*. This later collaborates with the micro-services *TNPMs* (Team Node Performance Micro-service) which are deployed on each node in order to define the performance index of all the nodes of the distributed system, and their loads index. So, the Load Balancer Ms will get the set of TNPMs micro-services from the DF Ms micro-service, and execute the performance test in collaboration with TNPMs micro-services in order to define the required metadata for elaborating the load balancing strategy (Fig. 8) according to these three global steps:

- **Initial Performance Test of nodes** The Ms Load Balancer executes the performance test on the node N_0, and then it sends the data D_0 to the TNPMs micro-services at t_0. Each TNPMs micro-service performs the performance test on its data D_0 and sends the result R_i that is composed by (Computation Time *Tpi*, and the number of CPUs NC_i), to the Ms Load Balancer at t1(i).

These results will be used by the Ms Load Balancer in order to get the metadata {Execution Time TE (TEi=(t1(i)-t0)), and Communication Latency TL (TLi= TEi-Tpi} needed to define the initial performance index and the loads index respectively according to the following equations:

$$NPI_i^{Ms_0} = \frac{MIN(\mathrm{TE}_i^{Ms_0})}{\mathrm{TE}_i^{Ms_0}} \tag{1}$$

$$\mathrm{NLI}_i^{Ms_0} = \mathrm{round}\left(\frac{\mathrm{nbMs}}{\mathrm{n}} \times \frac{\mathrm{NPI}_i^{Ms_0}}{\overline{\mathrm{NPI}_\imath^{Ms_0}}}\right) \tag{2}$$

$$\overline{NPI_\imath^{Ms_0}} = \frac{\sum_{i=0}^{n-1} NPI_i^{Ms_0}}{n} \tag{3}$$

- **Performance Index of the Nodes NPI** The Load Balancer Ms uses the metadata of the initial performance test {TE, TL, C_0} and the metadata of MsVPUs {C_k the complexity of service, and Z_k the amount of data exchanged between the node N_0 and N_i} in order to define the performance index NPI_i of each node N_i by :

$$NPI_i = \frac{MIN(\mathrm{TE}_i^{Ms})}{\mathrm{TE}_i^{Ms}} \tag{4}$$

Also, the execution time, and the latency of communication and the computational time can be estimated respectively by :

$$Tp_i^{Ms} = \frac{Tp_i^{Ms_0} \times C_k}{C_0} \tag{5}$$

$$TL_i^{Ms} = \frac{TL_i^{Ms_0} \times Z_k}{Z_0} \tag{6}$$

$$TE_i^{Ms} = (Tp_i^{Ms}/NC_i) + TL_i^{Ms} \tag{7}$$

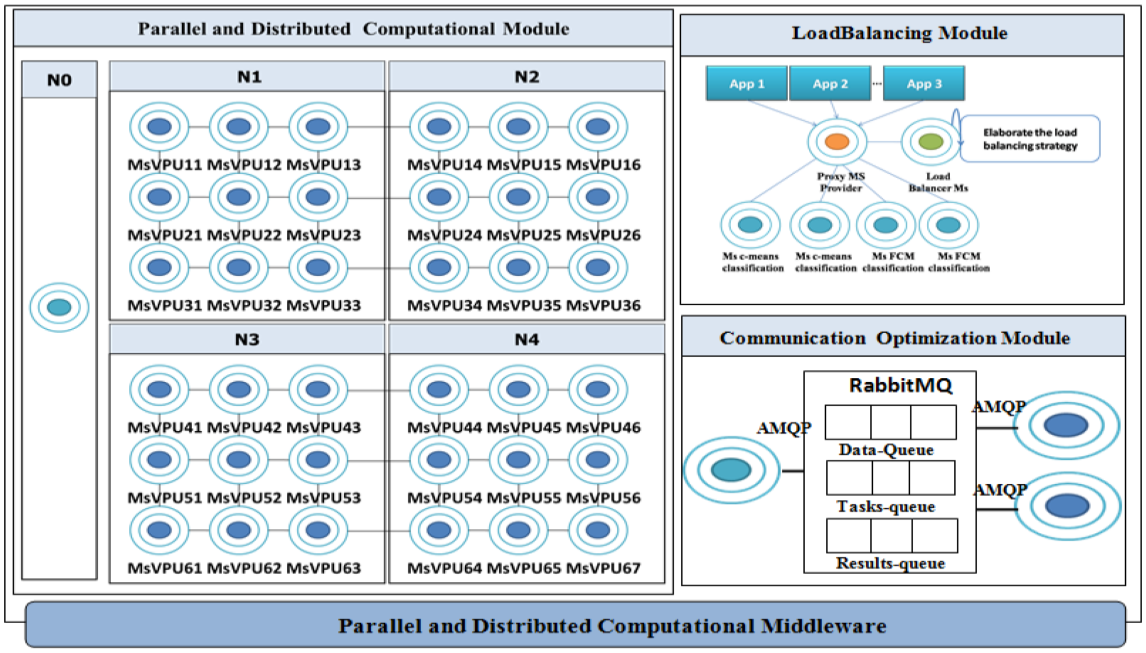


Fig. 6. Parallel and distributed computing middleware based micro-services modules.

- **Load Index of the Node NLI:** The Load Balancer Ms computes the load index of each node N_i by (8) based on the total number nbMs of micro-services needed for performing the request, and the number n of nodes, and the performance index NPI. This micro-service can get the micro-services information (address of node, port number) in order to choose the appropriate micro-services on each node N_i for performing the application request, by the way to maintain a balanced virtual machine.

$$NLI_i = round\ \left(\frac{nbMs}{n} \times \frac{NPI_i}{\overline{NPI_i}}\right) \qquad (8)$$

where
$$\overline{NPI_i} = \frac{\sum_{i=0}^{n-1} NPI_i}{n} \qquad (9)$$

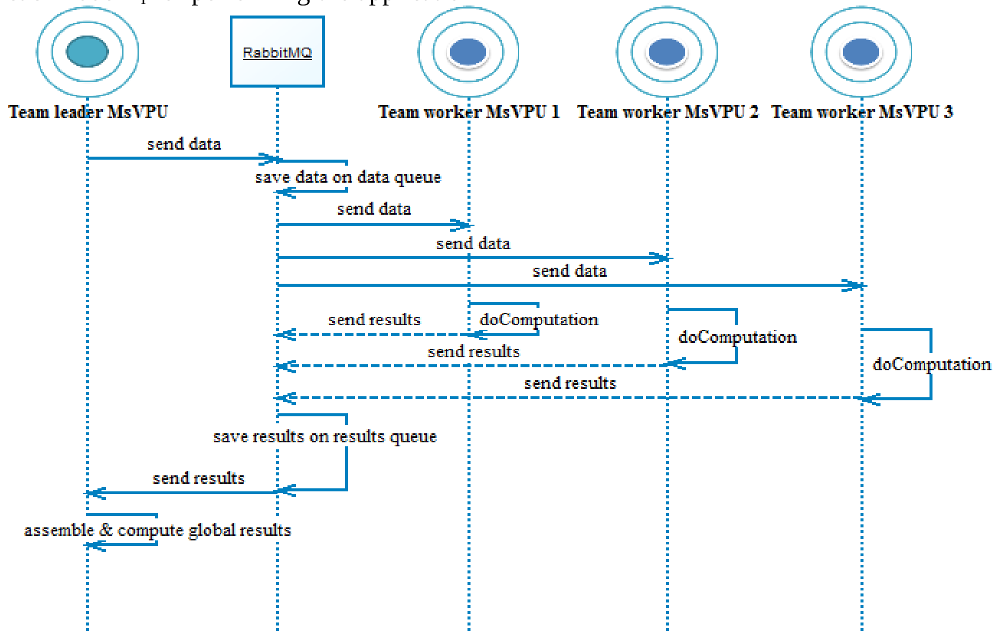


Fig. 7. Communication diagram of MsVPUs based asynchronous communication mechanism.

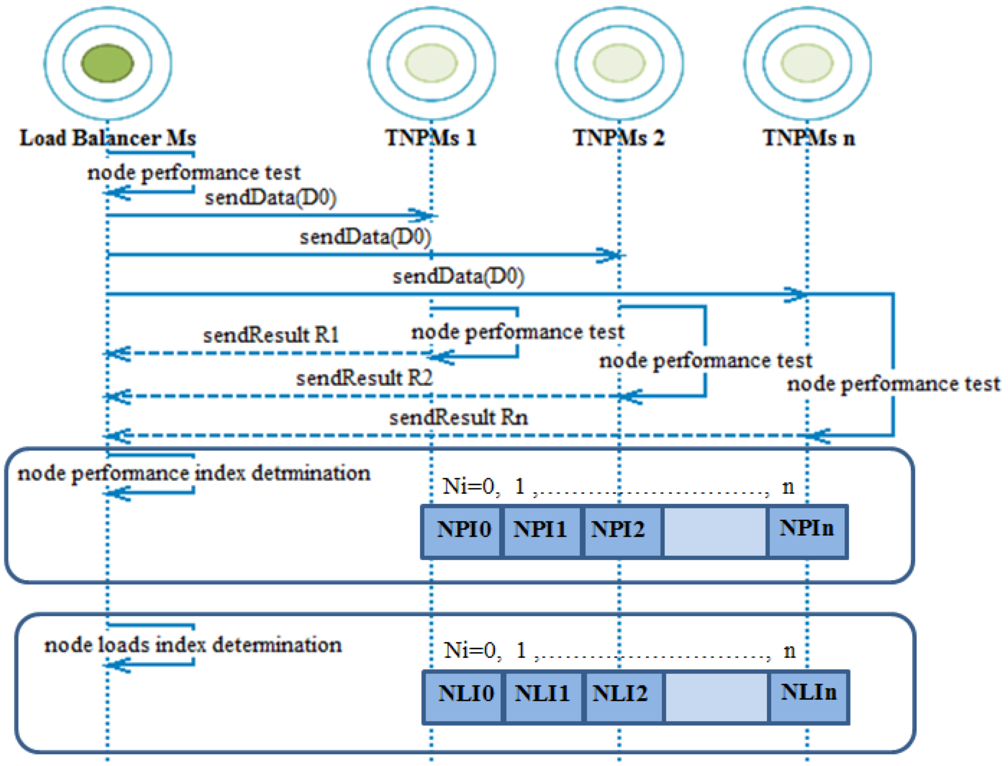


Fig. 8. Load balancing strategy based micro-services model.

IV. Application & Results

For testing the scalability and efficiency of the proposed virtual machine model, the two well-known SPMD classification algorithms; c-means [12] and Fuzzy c-means [13] are implemented as distributed services using the Spring Cloud Middleware.

A. *Distributed Implementation*

The classification algorithms are implemented on the MsVPUs of the computing model according to the communication diagram in Fig. 9. This diagram presents the micro-services MsVPUs and their implemented services in order to perform the classification of big image. For example, in order to perform the classification of the image the c-means algorithm is implemented according to distributed implementation DSCM (Distributed Service C-means) as follows:

- The Team leader MsVPU divides the input image on $NS = me \times ne$ elementary images.
- The Team leader MsVPU sends the elementary images NS to the Team workers MsVPUs, one per team worker MsVPU(s).
- Each Team worker MsVPU(s) gets its elementary image EI, and performs its classification service.
- For each iteration t

{

1) The Team leader MsVPU sends the initial class centers to all the Team workers MsVPU(s).

2) Each Team worker MsVPU(s) gets the class centers values and performs the classification service (doClassificationService). This service allows the Team worker MsVPU(s) to perform the classification on its elementary image and computes and elementary results :

ER2(s,k) the sum of colors of each class centers c_k, which is computed by:

$$ER2\ (s,k)=\sum_{j,x_j \in C_k} x_j \qquad (11)$$

ER3(s,k) the sum of the membership matrix of each class centers ck, which is computed by:

$$ER3(s,k)=\sum_{j=1}^{pi} c_k \qquad (12)$$

where **pi** is the number of pixels of the Team worker MsVPU elementary image EI.

ER1(s) the sum of distances of each class centers c_k, which is computed by:

$$ER1\ (s)= \sum_{j,x_j \in C_k} d(x_j, c_k) \qquad (10)$$

At the end of the classification, each Team worker MsVPU(s) sends its elementary results ER1(s), ER2(s, k), ER3(s,k) to its Team leader MsVPU.

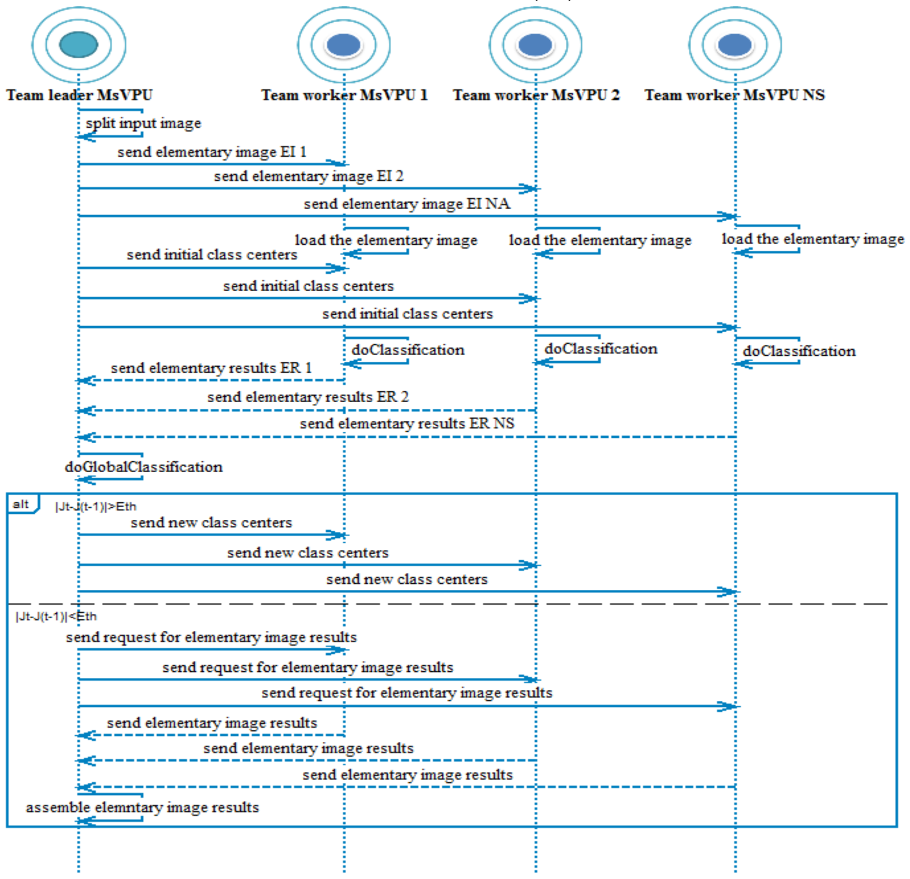

Team leader MsVPU
Team worker MsVPU 1
Team worker MsVPU 2
Team worker MsVPU NS
split input image
send elementary image EI 1
send elementary image EI 2
send elementary image EI NA
load the elementary image
load the elementary image
load the elementary image
send initial class centers
send initial class centers
send initial class centers
doClassification
doClassification
doClassification
send elementary results ER 1
send elementary results ER 2
send elementary results ER NS
doGlobalClassification
alt
|Jt-J(t-1)|>Eth
send new class centers
send new class centers
send new class centers
|Jt-J(t-1)|<Eth
send request for elementary image results
send request for elementary image results
send request for elementary image results
send elementary image results
send elementary image results
send elementary image results
assemble elemntary image results

Fig. 9. Communication diagram of distributed big data classification model main components.

3) The Team leader MsVPU gets the elementary results of the Team workers MsVPUs and performs the global classification service (doGlobalClassificationService) which is based on performing the three following sub services :

Assembling the elementary results: When the Team leader MsVPU receives the elementary results (ER1(s), ER2(s,k), ER3(s,k)). This later computes the global results (GER1(k), GER2(k), GER3(k)), respectively by (13),(14),(15).

GER1(k) the global value of ER1(s) of all the Team workers MsVPUs.

$$\text{GER1(k)}=\sum_{s=1}^{NS} \text{ER1}(s) \quad (13)$$

GER2(k) the global value of ER2(s) of all the Team workers MsVPUs.

$$\text{GER2(k)}=\sum_{s=1}^{NS} \text{ER2}(s,k) \quad (14)$$

GER3(k) the global value of ER3(s) of all the Team workers MsVPUs.

$$\text{GER3(k)}= \sum_{s=1}^{NS} \text{ER3}(s,k) \quad (15)$$

Calculate the new class centers: The Team leader MsVPU computes the new value of class centers based on the value of GER2(k) and GER3(k) by (16).

$$c_k = \frac{\text{GER2(k)}}{\text{GER3(k)}} \quad (16)$$

Computes the objective function J_t: The Team leader MsVPU uses the computed value of GER1(k) to determine the objective function by (17).

$$J_t = \sum_{k=1}^{c} \text{GER1}(k) \quad (17)$$

4) Test of convergence of the algorithm ($|J_t-J_{(t-1)}|<E_{th}$). The Team leader MsVPU compare the difference between the obtained objective function J_t and the one obtained in the previous iteration with the error (E_{th}), if $|J_t-J_{(t-1)}|<E_{th}$ (end), else (repeat from step 1 with the new value of the class centers).

}// End of iteration t

- The Team leader MsVPU requests the segmented elementary output images from the Team worker MsVPUs in order to assemble and provide the c outputs images and the final results to the application by Proxy provider Ms.

B. Results

The scalability and the performance of the proposed model are illustrated through an SPMD application. This application has to process a satellite image of size (row, column)=(7280, 7750) pixels on three output images C1, C2, C3 as shown in Fig. 10. The two classification services; c-means and fuzzy c-means using the same initial class centers (1.2, 2.5, 3.8) are performed under this application. We conclude in Table 1 and Table 2, that the two services; DSCM and DSFCM converge dynamically to the same final class centers (4.866, 112.396, 163.370). Fig. 11 and 12 show the dynamic convergence and the error of the objective function of both services.

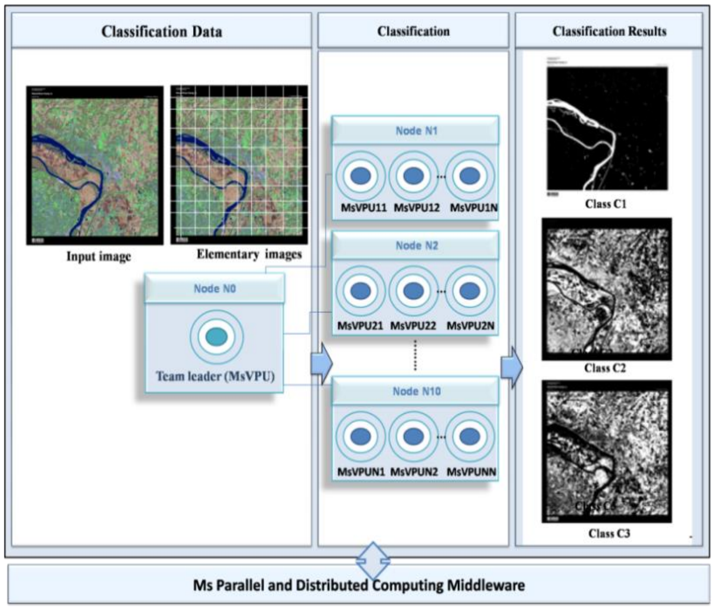


Fig. 10. Output classification image results using the proposed virtual machine based middleware.

TABLE I. DIFFERENT STATES OF THE DISTRIBUTED FUZZY C-MEANS SERVICE (DSCM) STARTING FROM THE CLASS CENTERS (C1, C2, C3) = (1.2,2.5,3.8).

Iteration	**Value of each class center**			**Absolute value of threshold**
	C_1	**C_2**	**C_3**	**$\|J_t-J_{t-1}\|$**
1	1,200	2,500	3,800	7,50E+00
2	0,001	2,253	132,119	1,27E+02
3	0,001	31,030	138,600	3,53E+01
4	0,219	47,574	140,414	1,86E+01
5	1,225	65,062	142,384	2,05E+01
6	3,481	87,135	145,767	2,77E+01
7	4,089	99,093	152,041	1,88E+01
8	4,379	105,239	156,876	1,13E+01
9	4,563	108,870	160,110	7,05E+00
10	4,706	110,646	161,733	3,54E+00
11	4,784	111,797	162,822	2,32E+00
12	4,866	112,396	163,370	1,23E+00
13	4,866	112,396	163,370	0,00E+00

For validating the performance of the proposed model, the classification time is analyzed for both services according to the involved number of MsVPUs in the classification in Fig. 13. We conclude that for both services the classification time achieves its minimum values of 26331 ms for DSCM and of 153970 ms for DSFCM using 32 MsVPUs. This number of MsVPUs is the required number for the classification of this case of image.

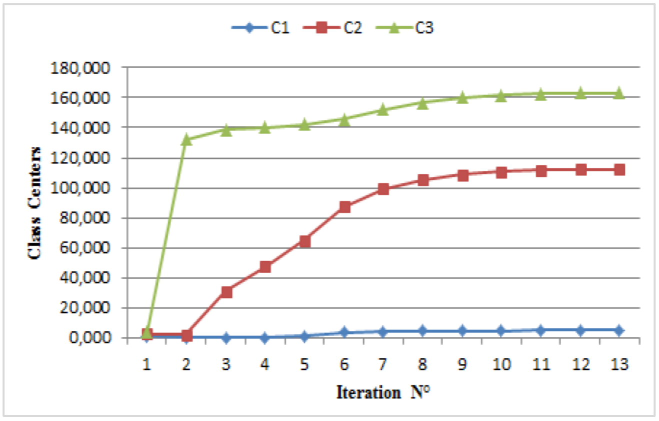
To illustrate the performance of the communication mechanism and the load balancing of this model, a compared study of the proposed model is performed with the mobile agent's model. In this study the corresponding application has to process 1000 elementary images of size (1024 ×786) pixels, by the way that 1000 MsVPU micro-services will execute the same service of complexity $C_k(x)=O(x^2)$ in parallel.

TABLE II. DIFFERENT STATES OF THE DISTRIBUTED FUZZY C-MEANS SERVICE (DSFCM) STARTING FROM THE CLASS CENTERS (C1, C2, C3) = (1.2,2.5,3.8).

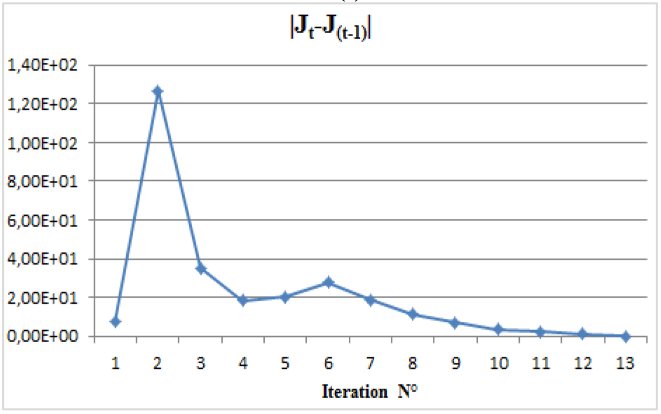
Iteration	Value of each class center			Absolute value of the error
	C_1	C_2	C_3	$\lvert J_n - J_{n-1}\rvert$
1	1,200	2,500	3,800	1,95E+09
2	56,561	123,684	128,646	1,03E+09
3	18,376	128,181	142,342	2,30E+08
4	6,348	122,692	153,589	1,37E+08
5	4,745	116,811	160,697	2,41E+07
6	4,344	114,478	163,436	1,28E+06
7	4,216	113,824	164,239	1,14E+06
8	4,180	113,666	164,460	4,42E+05
9	4,172	113,633	164,523	1,14E+05
10	4,170	113,630	164,543	3,39E+04
11	4,170	113,632	164,551	1,21E+04
12	4,170	113,634	164,555	5,26E+03
13	4,170	113,636	164,557	2,68E+03
14	4,170	113,637	164,558	1,51E+03
15	4,170	113,637	164,559	8,95E+02
16	4,170	113,638	164,559	5,42E+02
17	4,170	113,638	164,559	3,31E+02
18	4,170	113,638	164,560	2,03E+02
19	4,170	113,638	164,560	1,25E+02
20	4,170	113,638	164,560	7,69E+01
21	4,170	113,639	164,560	4,75E+01
22	4,170	113,639	164,560	2,91E+01
23	4,170	113,639	164,560	1,78E+01
24	4,170	113,639	164,560	1,11E+01
25	4,170	113,639	164,560	6,76E+00

From Fig. 14 and Table 3, it can be seen that the AVPU model achieves an acceptable load balancing with $\Omega_{MAX}^{EXP}(AVPU) - \Omega_{MIN}^{EXP}(AVPU)$ = 82,5517 s with the error ε_i(AVPU) ∈ [0.00014,0.03] and for the proposed model based MsVPU $\Omega_{MAX}^{EXP}(MsVPU) - \Omega_{MIN}^{EXP}(MsVPU)$ = 71,5271 s with ε_i(MsVPU) ∈ [0.00014, 0.03] which means a gain of performance of $\varphi 1 = \frac{\Omega_{MAX}^{EXP}(AVPU) - \Omega_{MIN}^{EXP}(AVPU)}{\Omega_{MAX}^{EXP}(MsVPU) - \Omega_{MIN}^{EXP}(MsVPU)} =$ 1,15413 compared to the AVPUs based model. This is due, to the lightweight communication mechanism of the micro-services compared to the mobile agents. So, the both models integrate the mechanism to ensure high performance computing.

From Table 4, it can be seen that the both models provide autonomous virtual computing units which enhance the processing power, and manage the computing challenges; heterogeneity of computing nodes and the message passing mechanism of computing units. So, the HPC applications can take advantages of these two models.

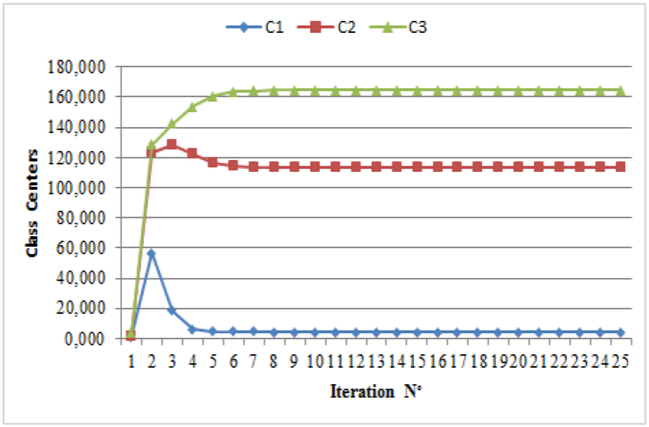


(a)

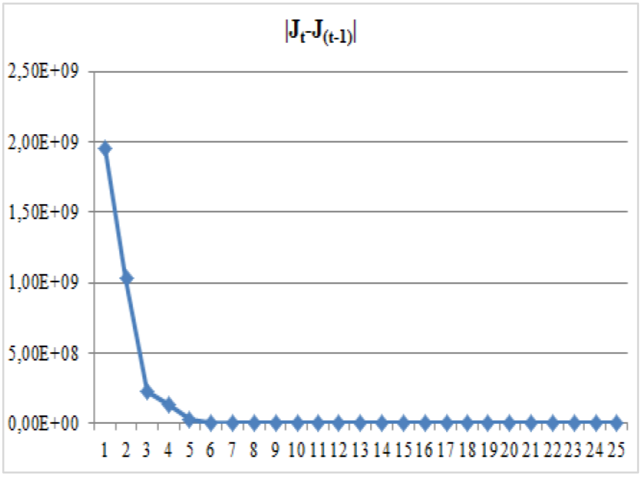


(b)

Fig. 11. Dynamic convergence of DSCM service with initial class centers (c1, c2, c3) = (1.2, 2.5, 3.8); (a) Class centers, (b) Error of the objective function.



(a)



(b)

Fig. 12. Dynamic convergence of DSFCM service with initial class centers (c1, c2, c3) = (1.2, 2.5, 3.8); (a) Class centers, (b) Error of the objective function.

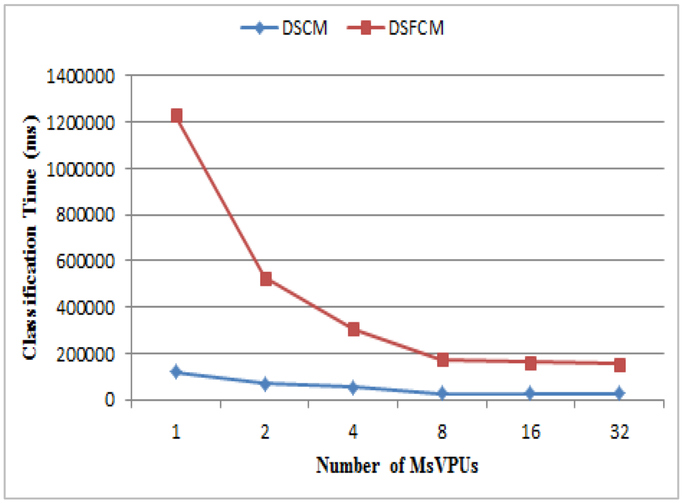


Fig. 13. Classification Time depending on the number of MsVPUs for c-means service DSCM and Fuzzy c-means service DSFCM.

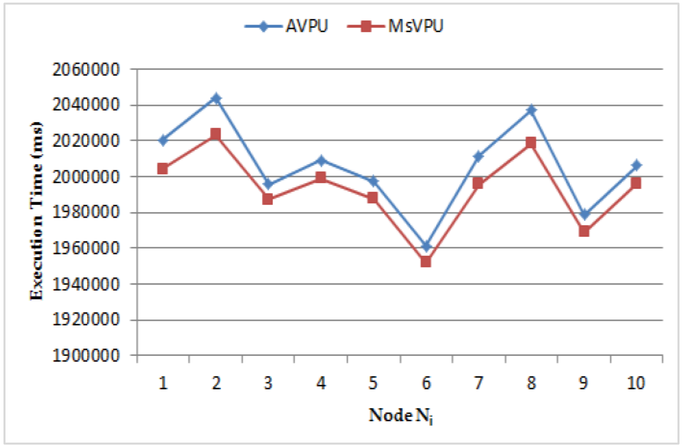


Fig. 14. Execution time for each node N_i, for the both computing models; Agent AVPU Model and Micro-service MsVPU Model.

TABLE III. COMPARISON OF EXECUTION TIME FOR EACH NODE N_i FOR THE BOTH COMPUTING MODELS; AGENT AVPU MODEL AND MICRO-SERVICE MSVPU MODEL

N_i	**AVPU Model**		**MsVPU Model**	
	Ω_i^{EXP} (ms)	ε_i	Ω_i^{EXP} (ms)	ε_i
0	2020513,677	0,007556677	1989694,791	0,007947277
1	2043923,005	0,019275406	2012147,181	0,01937828
2	1996003,738	0,004652171	1964820,136	0,004652169
3	2009285,591	0,001981942	1977301,264	0,001761339
4	1997420,298	0,003964798	1966256,613	0,003868472
5	1961371,238	0,021891009	1930789,454	0,021891011
6	2011818,492	0,003214181	1980248,984	0,003214181
7	2037281,587	0,015916015	2005008,187	0,015801354
8	1978621,624	0,013275925	1964954,997	0,004573581
9	2006293,217	0,000495296	1975019,785	0,000594413

TABLE IV. COMPARISON BETWEEN THE PARALLEL AND DISTRIBUTED COMPUTING MODELS; AGENT AVPU MODEL AND MICRO-SERVICE MSVPU MODEL.

	AVPU Model	**MsVPU Model**
Virtual Processing Unit	Mobile agent	Micro-service
Fault Tolerance	Agent clone mechanism	Micro-service plateform librairies. Example for Spring cloud (Netflix Hystrix)
Load Balancing	Agent migration	Micro-service plateform librairies. Example for Spring cloud (Eureka)
Communication	Asynchronous communication with ACL (Agent Communication Language) message.	Asynchronous communication with AMQP (Advanced Message Quering Protocol) Protocol.
Deployment	With Multi agents platform using JVM	With Micro-service containers using Cloud Computing.
Performance	High	High

V. Related Works

Several inspired researches have been proposed for scaling up the cloud computing [14]-[18]. For example in [18], the authors presented a new approach for horizontally scaling cloud resources, and in [19] for load balancing, and resources scheduling [20] in cloud. In [21] the authors presented the cloud concept and its emerged services that deal with the IoT trends, and they notice also that the applications with complex data-intensive computations are the best candidate to take advantages of cloud computing. Therefore, by applying the parallel and distributed simulation on cloud, the performance of these applications depends on the applied synchronization algorithms [22]. Our approach considers the performance of the HPC applications which are implemented on cloud architectures using micro-services, and deals with intensive computing units communication.

The cloud native applications [23], [24] with their related micro-services architectures can be promising methodologies for HPC applications in cloud computing. For example in [25] the authors presented a performance evaluation of micro-services architectures using containers: master-slave and nested-container, and in [26] they discussed the benefit of implementing micro-services architecture for emerging the telecom application. Also, the micro-services approach is implemented to digital curation infrastructure by devolving function into a set of micro-services which grants the deployment flexibility and simplify of the development and the maintenance [27]. These features of the micro-services architectures deal with the new trends of the HPC middleware [5], and ensure the scalability of the distributed applications [28].

VI. Conclusion

The massively distributed virtual machine model based micro-services for HPC. This model integrates a cooperative team of micro-services that are deployed as virtual computing components MsVPUs (Micro-service Virtual Processing Units) for performing the parallel programs as services according to different architectures and topologies. The MsVPUs are the virtual processors that enhance the processing power. Also, they use the asynchronous communication mechanism based AMQP protocol to optimize the communication cost of the model. This model implements a load balancing module for managing the micro-services and ensures the high performance computing. In this paper, the efficiency and the performance of this model are illustrated through an application of classification using the two services; c-means and Fuzzy c-means. In each node a specific number of MsVPUs are deployed according to the load balancing method. So, the MsVPUs cooperate in different nodes and execute the application by the way to ensure a balanced virtual machine with low communication cost. Compared to the mobile agents based model, the proposed model grants a lightweight communication mechanism which optimizes significantly the communication cost. Also, the proposed virtual machine capabilities provides the ability to extended its model to an elastic platform that will be deployed in Cloud as PaaS (Platform as a Service).

References

[1] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, “Big Data computing and clouds: Trends and future directions,” J. Parallel Distrib. Comput., vol. 79, pp. 3–15, 2015.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, and RH, “Above the clouds: A Berkeley view of cloud computing,” Univ. California, Berkeley, Tech. Rep. UCB , pp. 07–013, 2009.

[3] M. Zakarya and L. Gillam, “Energy efficient computing, clusters, grids and clouds: A taxonomy and survey,” Sustain. Comput. Informatics Syst., vol. 14, pp. 13–33, 2017.

[4] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, and M. Schulz, “Exploiting Redundancy and Application Scalability for Cost-Effective, Time-Constrained Execution of HPC Applications on Amazon EC2,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 9. pp. 2574–2588, 2016.

[5] C. Engelmann, H. Ong, and S. L. Scott, “Middleware in Modern High Performance Computing System Architectures,” in Computational Science -- ICCS 2007: 7th International Conference, Beijing, China, May 27 - 30, 2007, Proceedings, Part II, Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 784–791.

[6] H. El-Rewini and M. Abd-El-Barr, Advanced Computer Architecture and Parallel Processing. 2005.

[7] R. Miller and Q. F. Stout, “Geometric Algorithms for Digitized Pictures on a Mesh-Connected Computer,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 7, no. 2, pp. 216–228, 1985.

[8] J. Wu, J. JaJa, and E. Balaras, “An Optimized FFT-Based Direct Poisson Solver on CUDA GPUs,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 3, pp. 550–559, 2014.

[9] A. Rafique, G. A. Constantinides, and N. Kapre, “Communication Optimization of Iterative Sparse Matrix-Vector Multiply on GPUs and FPGAs,” IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 1, pp. 24–34, Jan. 2015.

[10] M. Youssfi, O. Bouattane, and M. O. Bensalah, “A Massively Parallel Re-Configurable Mesh Computer Emulator: Design, Modeling and Realization,” J. Softw. Eng. Appl., pp. 11–26, 2010.

[11] M. Youssfi, O. Bouattane, F. Z. Benchara, and M. O. Bensalah Mohammed, “A Fast Middleware For Massively Parallel And Distributed Computing,” IJRCCT, vol. 3, no. 4, 2014.

[12] O. Bouattane, B. Cherradi, M. Youssfi, and M. O. Bensalah, “Parallel c-means algorithm for image segmentation on a reconfigurable mesh computer,” Parallel Comput., vol. 37, no. 4, pp. 230–243, 2011.

[13] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[14] S. Niu, J. Zhai, X. Ma, X. Tang, W. Chen, and W. Zheng, “Building Semi-Elastic Virtual Clusters for Cost-Effective HPC Cloud Resource Provisioning,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 7, pp. 1915–1928, 2016.

[15] B. Mao, S. Wu, and H. Jiang, “Exploiting Workload Characteristics and Service Diversity to Improve the Availability of Cloud Storage Systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 7. pp. 2010–2021, 2016.

[16] W. Xiao, W. Bao, X. Zhu, C. Wang, L. Chen, and L. T. Yang, “Dynamic Request Redirection and Resource Provisioning for Cloud-Based Video Services under Heterogeneous Environment,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 7. pp. 1954–1967, 2016.

[17] H. Wang, Z. Kang, and L. Wang, “Performance-Aware Cloud Resource Allocation via Fitness-Enabled Auction,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 4. pp. 1160–1173, 2016.

[18] D. Grimaldi, A. Pescape, A. Salvi, s. santini, and V. Persico, “A Fuzzy Approach based on Heterogeneous Metrics for Scaling Out Public Clouds,” IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99. p. 1, 2017.

[19] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, “A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2. pp. 305–316, 2016.

[20] X. Xu, L. Cao, X. Wang, X. Xu, L. Cao, and X. Wang, “Resource pre-allocation algorithms for low-energy task scheduling of cloud computing,” Journal of Systems Engineering and Electronics, vol. 27, no. 2. pp. 457–469, 2016.

[21] S. Sharma, V. Chang, U. S. Tim, J. Wong, and S. Gadia, “Cloud-based emerging services systems,” International Journal of Information Management, 2016.

[22] G. D’Angelo and M. Marzolla, “New trends in parallel and distributed simulation: From many-cores to Cloud Computing,” Simul. Model. Pract. Theory, vol. 49, pp. 320–335, 2014.

[23] N. Kratzke and P. C. Quint, “Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study,” J. Syst. Softw., vol. 126, pp. 1–16, 2017.

[24] D. Namiot and M. Sneps-Sneppe, “On Micro-services Architecture,” Int. J. Open Inf. Technol., vol. 2, no. 9, 2014.

[25] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar, and M. Steinder, “Performance Evaluation of Microservices Architectures Using Containers,” 2015 IEEE 14th International Symposium on Network Computing and Applications. pp. 27–34, 2015.

[26] M. Sneps-Sneppe and D. Namiot, “Micro-service Architecture for Emerging Telecom Applications,” Int. J. Open Inf. Technol., vol. 2, no. 11, 2014.

[27] S. Abrams, J. Kunze, and D. Loy, “An Emergent Micro-Services Approach to Digital Curation Infrastructure,” Int. J. Digit. Curation, vol. 5, no. 1, 2010.

[28] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, “Microservices: How To Make Your Application Scale,” 2017.