Journal of Software Engineering and Applications, 2015, 8, 103-113
Published Online March 2015 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2015.83011

Distributed C-Means Algorithm for Big Data Image Segmentation on a Massively Parallel and Distributed Virtual Machine Based on Cooperative Mobile Agents

Fatéma Zahra Benchara[1], Mohamed Youssfi[1], Omar Bouattane[1], Hassan Ouajji[1], Mohammed Ouadi Bensalah[2]

[1]Laboratory SSDIA, ENSET Mohammedia, University Hassan II of Casablanca, Casablanca, Morocco
[2]FSR, University Mohammed V, Rabat, Morocco
Email: benchara.fatemazahra@gmail.com, med@youssfi.net, o.bouattane@gmail.com

Received 11 February 2015; accepted 28 February 2015; published 3 March 2015

Abstract

The aim of this paper is to present a distributed algorithm for big data classification, and its application for Magnetic Resonance Images (MRI) segmentation. We choose the well-known classification method which is the c-means method. The proposed method is introduced in order to perform a cognitive program which is assigned to be implemented on a parallel and distributed machine based on mobile agents. The main idea of the proposed algorithm is to execute the c-means classification procedure by the Mobile Classification Agents (Team Workers) on different nodes on their data at the same time and provide the results to their Mobile Host Agent (Team Leader) which computes the global results and orchestrates the classification until the convergence condition is achieved and the output segmented images will be provided from the Mobile Classification Agents. The data in our case are the big data MRI image of size (m × n) which is splitted into (m × n) elementary images one per mobile classification agent to perform the classification procedure. The experimental results show that the use of the distributed architecture improves significantly the big data segmentation efficiency.

Keywords

Multi-Agent System, Distributed Algorithm, Big Data Image Segmentation, MRI Image, C-Means Algorithm, Mobile Agent

How to cite this paper: Benchara, F.Z., Youssfi, M., Bouattane, O., Ouajji, H. and Bensalah, M.O. (2015) Distributed C-Means Algorithm for Big Data Image Segmentation on a Massively Parallel and Distributed Virtual Machine Based on Cooperative Mobile Agents. *Journal of Software Engineering and Applications*, **8**, 103-113.
http://dx.doi.org/10.4236/jsea.2015.83011

1. Introduction

Multi-agent system is the aim of several researchers and developers in order to design and implement some methods and solutions which improve existing methods and create new ones in order to achieve high performance results, save time and make the work easy in many domains. As examples, in the renewable energy domain [1] the authors proposed a method that parallelizes the procedure for detection of illegal consumers of electricity which can be improved by the use of a distributed architecture based on the multi-agent systems; and in web semantic domain [2] the authors proposed a novel model for automatic construction of business processes based on multi-agent systems. In the high performance computing domain, the authors in [3] proposed the use of this technology in order to improve the management, the flexibility and the reusability of grid like parallel computing architectures.

For each new technology or method, we can distinguish its straightness and weakness. That is the case of the researchers who start by analyzing these two items related to their field in order to propose and argue their ideas. As in [4], we start by analyzing the state of the art about the parallel computing strategies. We can say that the growth of number of data to be processed and the convergence of the applications from the standard to the distributed one make it necessary to use many cores of processors as a supercomputer. Due to the high cost of the supercomputers and the real parallel machines, it is feasible to use the parallel virtual machine as in [5] for specific work and [6] for global needs. It is based on a parallel and distributed grid computing that cooperates with the power of processors from heterogeneous computers by forming a distributed system for parallel computing. So it is possible to split tasks and data between them in order to reach the hoped level of high performance computing. But we do not need to forward the main component which is responsible of creating and managing the grid computing and also which supports the computing performance keys (load balancing, fault tolerance and communication cost) in the distributed system. As in [7] we propose a middleware for parallel and distributed computing which manipulates virtual processors (VPU objects) in different nodes of the grid. It is an innovated idea that was developed using the power of the multi-agent system.

In this paper, we propose a new method for big data classification which is a distributed c-means algorithm. It is based on different previous works and applied to MRI cerebral segmentation images in order to be implemented on a mobile agents' parallel and distributed virtual machine. In this case we use the power of multi-agent system especially the mobile agents to answer to the parallel computing challenges where some of them are presented in [8] with a critical way. So we need to use applications that can run the scalable parallel algorithms and ensure a high performance computing for different cases. As in [9] the authors showed the improvement of the time efficiency of a medical reasoning system using multi-agent architecture. In [10] the authors proposed a parallel and distributed model based on mobile agent for high performance computing where we implemented the distributed c-means algorithm for MRI cerebral images segmentation.

Big data image segmentation is an analyzing image strategy which is emerged especially in the medicine domain using different algorithm specifications where the execution of the programs is based especially on the parallel computing strategy in order to achieve effective results. In this case we choose to test by the c-means classification algorithm to implement it on different parallel architectures. In [11], the authors proposed a method that subdivided the c-means program into several elementary operations which were organized in pipeline structure. In [12], the authors have proposed an optimal clustering solution based on a reconfigurable array of processors, while in [13] the authors proposed a SIMD parallel segmentation c-means algorithm.

The proposed method in this paper is a distributed c-means algorithm for big data classification; it is assigned to be implemented on a parallel and distributed virtual machine based on mobile agents. The corresponding distributed program is validated on a multi-agent system platform using the JADE middleware and an extensible and flexible smart grid computing of the same size as the input image. This paper is organized as follows: the second section makes an overview of the distributed computational model. Section 3 presents the distributed c-means algorithm while Section 4 is focused on the results concluded by implementing this distributed c-means algorithm. The last section presents some interesting obtained results and the effectiveness, and some perspectives of this work.

2. Distributed Computational Model

2.1. Model Overview

A cooperative multi-agent platform is a massively parallel and distributed virtual machine where the mobile

agents which represent the Processing Elements (PEs) are involved in the execution of the parallel programs on different structures (SIMD, SPMD, MIMD…) and topologies (2D Mesh, 3D Mesh, hierarchical…) according to the parallel problem to resolve.

In this work we implement the well known algorithm for big data image segmentation which is the c-means algorithm as a distributed program on this model by using SPMD structure. The main components which are engaged to perform the execution of this program are the host agent and the team worker agents that are designed to play the roles introduced in **Figure 1**.

2.2. Intelligent Mobile Agents Grid Computing

This model is based on an intelligent grid computing organized as a bidimensional 2D mesh topology (**Figure 2**) of size m × n, in which each mobile agent MPE (Mobile Processing Element) is localized in row *i* and column *j* and has an AID (Agent Identifier). **Figure 2** shows an example of (2 × 2) workers machines hosted by M0. All the agents (*i*, *j*) in the mesh execute their tasks autonomously and maintain asynchronous communication by exchanging ACL (Agent Communication Language) Messages between each other over the grid computing. In order to achieve a high performance computing of the big data image segmentation, it is important to deploy the large amount of skills of these agents, as:

1) Autonomy: each mobile agent can perform, image classification autonomously. It can extend its activities to make the diagnostic on images and generates reports and data for eventual decisions.

2) Adaptability: it can execute its tasks in different environments with the same performance as in the environment where it was created.

3) Mobility: it can move from one machine to another and resume its tasks if some parallel computing challenges happen.

4) Asynchronous communication ability: it can exchange data, tasks and knowledge using ACL message in remote or local communication.

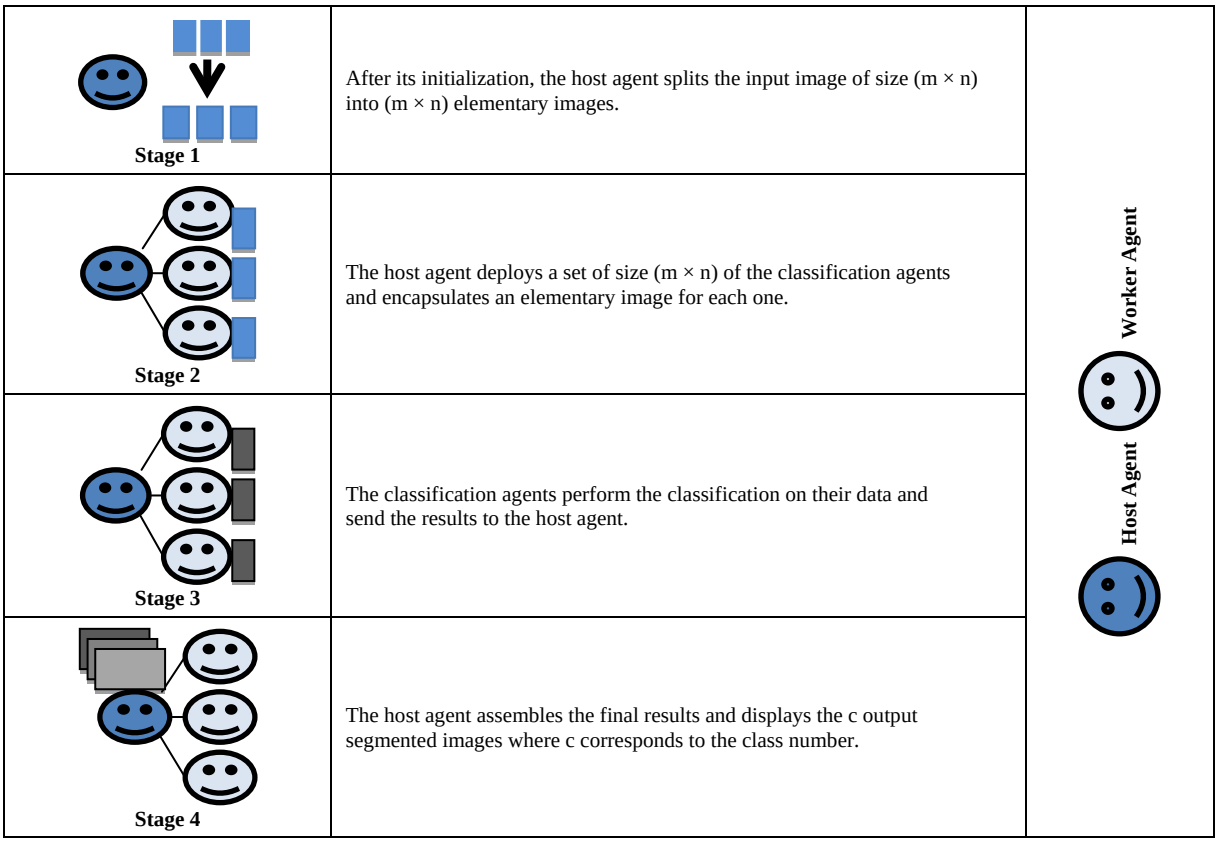


Figure 1. Overview of the main goal of the model.

3. Distributed Segmentation Algorithm

3.1. Standard C-Means Algorithm

The c-means algorithm as defined in [13], is an image segmentation algorithm which consists on a partitioned groups of set S of n attribute vectors into c classes (clusters C_i i = 1, ⋯, c). In this algorithm [13] authors used the pixel gray level as the main classification attribute. The finale goal is to find the class centers (centroïd) that minimize the cost function J defined by the following equation: $J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{k \in Ci} d\left(x_k, c_i\right)$. To do so, we have to execute the different equations, given in **Table 1**.

The c-means classification is achieved using the following algorithm stages which are summarized in **Figure 3**.

3.2. Distributed C-Means Algorithm

In this section, we present our distributed implementation of the c-means algorithm on a cooperative multi-agent platform using the 2D mesh architecture of the same size as the input image (MRI cerebral image of 256 gray levels). MRI images are stored on a cooperative multi-agent platform one pixel per agent. In order to perform the distributed c-means classification of the image in this model we need to perform the following three global stages as shown in **Figure 4**.

1) Host agent initialization: when our segmentation platform goes on. The host agent is the first one created, and initialized by the size of the data image which is in our case (m × n).

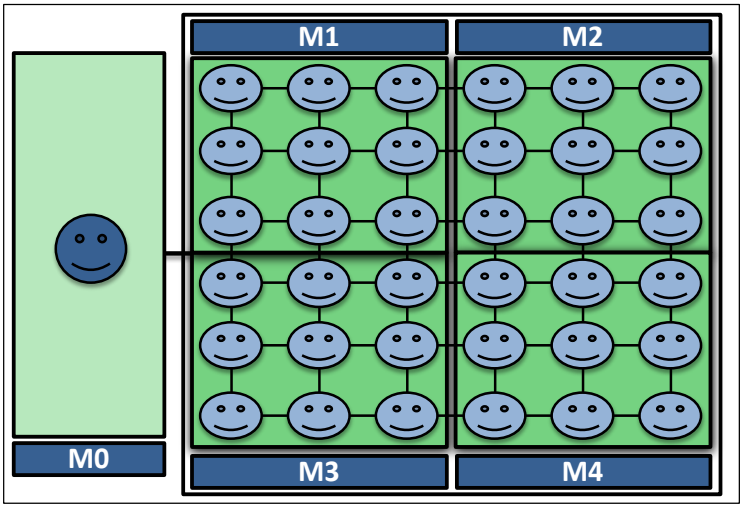
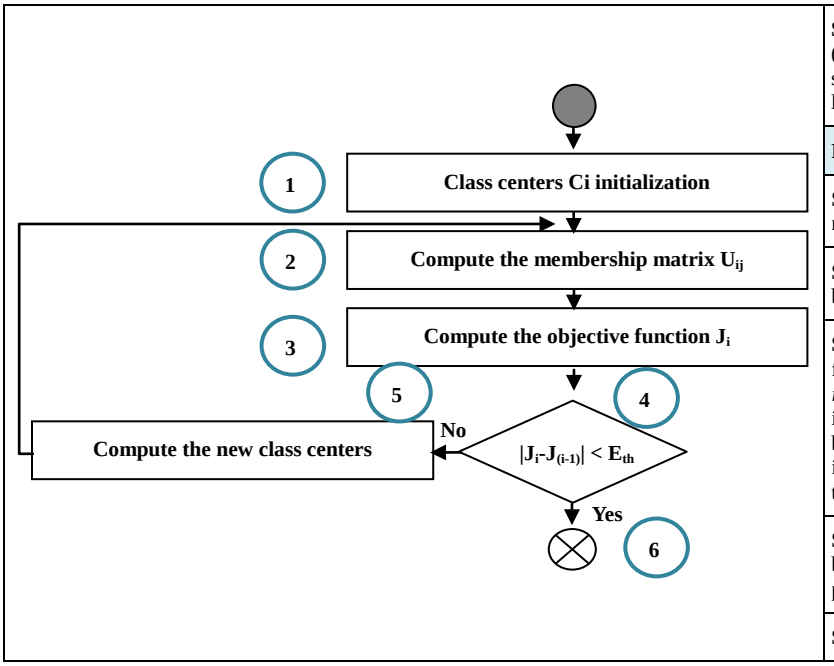


Figure 2. 2D mesh mobile agent grid computing of 5 physical machines.

Table 1. The different equations needed for the classification c-means.

Equation number	Equation	Description
1	$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k \in Ci} \left\| x_k - c_i \right\|^2 \right)$	The objective function defined by the Euclidean distance (distance between i^{th} center c_i and the k^{th} data of S).
2	$u_{ij} = \begin{cases} 1 \; if \; \left\| x_k - c_i \right\|^2 \le \left\| x_k - c_k \right\|^2, \forall k \ne i, \\ 0 \quad Otherwise \end{cases}$	The membership matrix element where i = 1 to c, j = 1 to n, for a total number n of points in S. It has two properties: $\sum_{i=1}^{c} u_{ij=1,} \forall j = 1, \cdots, n$ $\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n$
3	$c_i = \frac{1}{\left\| C_i \right\|} \sum_{k,(x_k \in C_i)} x_k$	The class center where i = 1 to c by the average of all its attribute vectors and $\left\| C_i \right\|$ is the cardinal of C_i.



Stage 1: Initialize the class centers C_i (i = 1, ⋯, c). This is carried out by selecting randomly c points in the gray level scale [0…255].
For each iteration *i*:
Stage 2: Determine the membership matrix using Equation (2).
Stage 3: Compute the cost function J by Equation (1).
Stage 4: Compare the obtained cost function J to the one computed at iteration $i-1$ and stop the loop (*i.e.* go to stage 6) if the absolute value of the difference between the successive cost functions is lower than an arbitrary defined threshold (Eth).
Stage 5: Calculate the new class centers by Equation (6), and return back to perform stages 2, 3 and 4.
Stage 6: End.

Figure 3. Standard classification c-means stages.

2) Grid construction: just after the initialization, the host agent splits the input image into (m × n) elementary images and then deploys a list of the classification agents which each of them encapsulates an elementary image and moves to a specific destination where it supposes to execute it tasks.

3) C-means classification: to start the data segmentation, the classification agents process, the classification tasks and the initial class centers which are sent by their host agent, and execute the local class determination task in order to compute the membership matrix. Then they replay the message with the results to their host agent which assembles the elementary results from all the classification agents and performs the global class determination and computes the cost function J and test the condition. If it is not achieved, it sends a new class centers to the classification agents to repeat the classification until it is achieved. Finally the host agent sends a request to the classification agents in order to send the output elementary images which will be assembled in order to display the c output segmented images for the classification.

4. Distributed C-Means Algorithm Implementation and Results

4.1. Multi-Agent System Middleware

This cooperative and intelligent platform is a distributed and parallel virtual machine on which we implemented the image segmentation algorithm according to the model of **Figure 5**. It is based especially on the JADE middleware [14] and on the mobile agent skills.

Thanks to the JADE middleware, we prepare the platform where we run the parallel and distributed c-means program. The main components of this model are:

- The main container: it is the first container which starts in the platform and which is different from the others by its two special agents (the AMS agent and the DF agent). It connects all the performed containers in the platform.
- The host container: is the second container that is started in the platform where the host agent is created in order to manage the required infrastructure and computational inside in the physical grid computing.
- The agent containers: the free containers are started in the platform to receive groups of the classification agents so that each group can execute its classification tasks.

In a multi-agent system, we distinguish the mobile agents that have interesting and additional skill from the other agents which is the ability to migrate between nodes. It is considered the one key of the high performance computing and it fit well with our main goal.

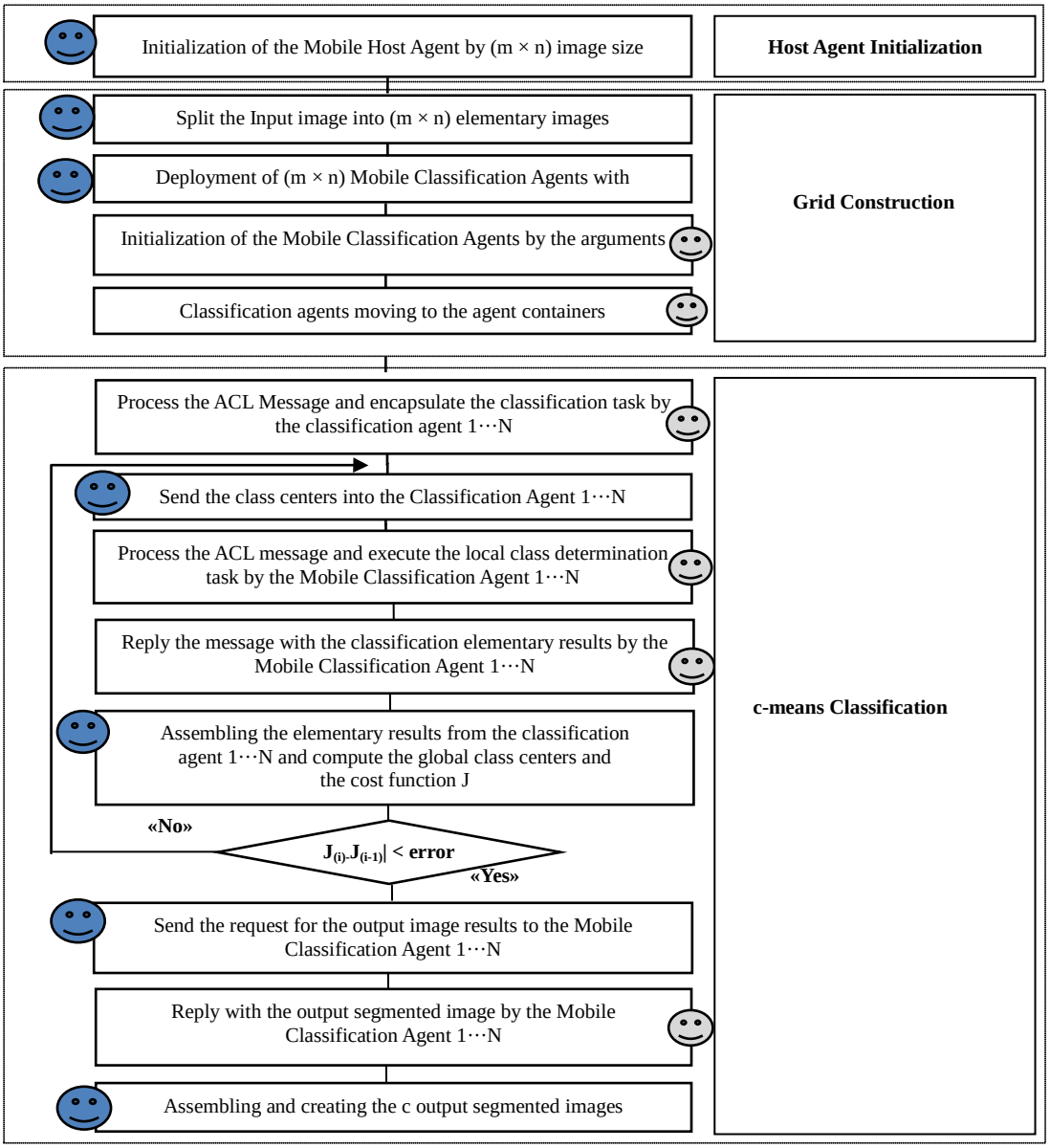


Figure 4. Distributed c-means program execution organization chart.

The mobile agents are considered as Processing Elements (PEs) which encapsulate tasks and data in order to achieve the parallel programs execution. So they are independent inside their containers, where they are deployed, in order to stay or move to the most suitable environment to achieve perfectly their tasks. They have the ability to communicate between each other asynchronously by exchanging ACL messages. Thanks to the high performance of the mobile agent we can take strong control about all the parallel computing challenges: load balancing, fault tolerance and communication cost. This allows us to constitute a cooperative and intelligent grid for high performance computing.

4.2. Implementation and Results

This part presents the implementation of the proposed distributed c-means program in our cooperative platform according to the organization chart presented in **Figure 4** using the MRI cerebral image. We obtain the following results: the **Figure 6(a)** corresponds to a human cerebral cut; it is the original input image of the program which is split into (m × n) elementary images as in **Figure 6(b)**. And the **Figures 6(c)-(e)**, are the segmented

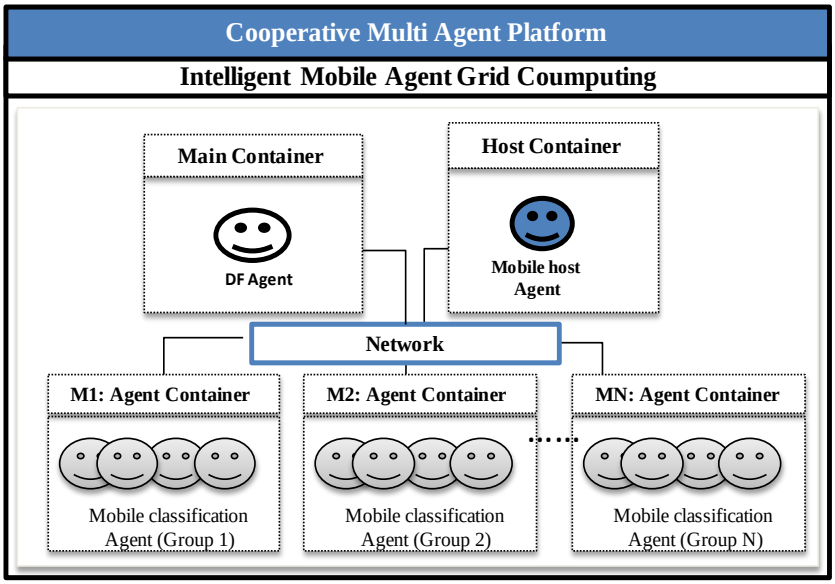


Figure 5. Distributed computational model architecture for c-means classification.

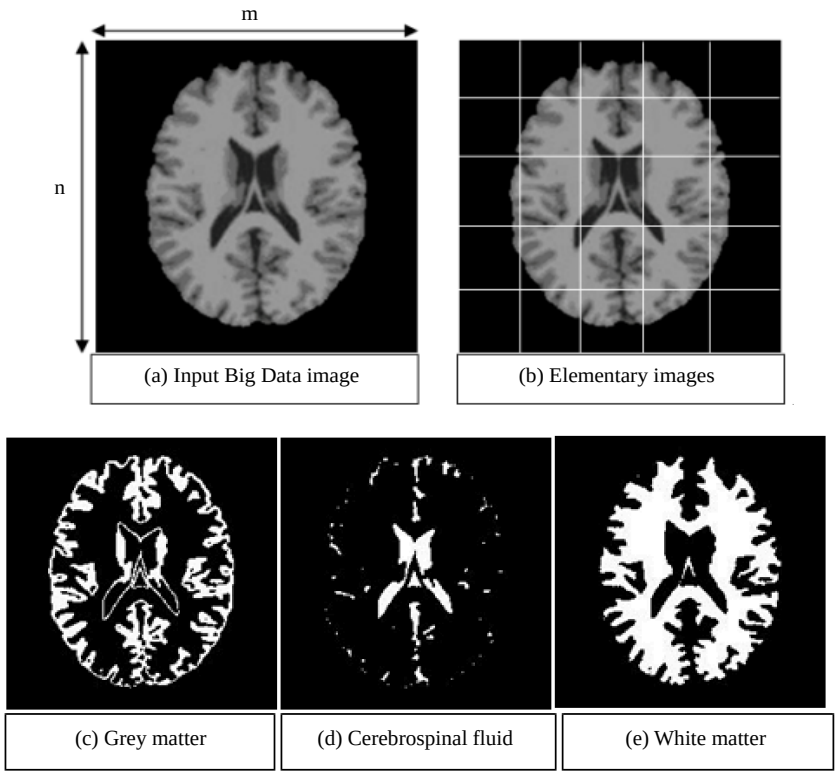


Figure 6. Segmentation results by the elaborated distributed program.

output images where each of them corresponds to a specific class: the gray matter, the cerebrospinal fluid and the white matter.

The effectiveness features of this distributed method for data image classification is shown under these three cases of studies for dynamic convergence analysis and time classification analysis under the fourth case.

1) $(c_1, c_2, c_3) = (1, 2, 3)$ where we see clearly in **Table 2** the convergence of the algorithm after 15 iterations to the final class centers $(c_1, c_2, c_3) = (28.60, 101.60, 146.03)$ and which are illustrated in **Figure 7**.

Table 2. Different states of the classification algorithm starting from class centers (c_1, c_2, c_3) = (1, 2, 3).

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	1.00	2.00	3.00	4.96E+06
2	1.00	2.00	120.14	4.42E+06
3	1.00	33.86	129.57	3.26E+06
4	4.38	51.91	132.04	1.33E+06
5	6.15	58.40	133.23	1.50E+05
6	7.14	60.05	133.43	5.36E+04
7	7.14	65.14	134.44	2.34E+05
8	16.70	72.44	134.64	9.02E+05
9	21.49	90.47	141.16	2.76E+06
10	24.68	93.37	142.41	1.24E+05
11	28.60	95.44	142.42	7.92E+04
12	28.60	96.71	142.98	1.02E+05
13	28.60	100.94	145.51	2.41E+05
14	28.60	101.60	146.03	5.64E+03
15	28.60	101.60	146.03	0.00E+00

(a) (b)

Figure 7. Dynamic changes of the class centers starting from (c_1, c_2, c_3) = (1, 2, 3). (a) Class centers; (b) Error of the objective function.

2) (c_1, c_2, c_3) = (1, 30, 255) where the algorithm converges to the same final class centers of the case 1 (c_1, c_2, c_3) = (28.60, 101.60, 146.03) after 5 iterations as shown in **Table 3** and which are illustrated in **Figure 8**.

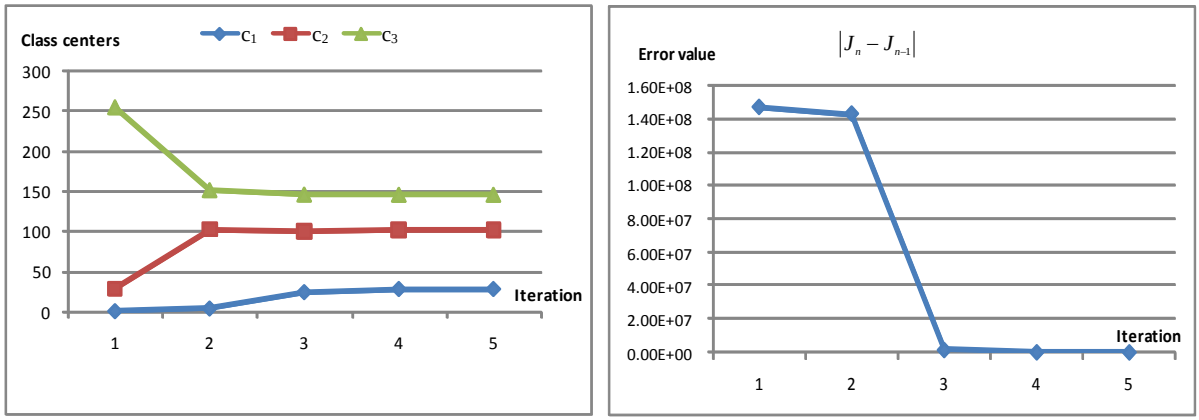
3) (c_1, c_2, c_3) = (140, 149, 255) where the algorithm converges to the final class centers (c_1, c_2, c_3) = (33.37, 103.50, 146.03) after 10 iterations as shown in **Table 4** which are illustrated in **Figure 9**.

In **Figure 10**, we present the variation of the classification time according to the number of agents involved in the computing using the initial class centers (1, 2, 3).

In this experience, the used distributed system is composed by 8 CPUs. We notice that the execution time is decreasing according to the number of classification agents. From 8 agents, the classification time saved become very small. This is due to the fact that the distributed system is composed of 8 CPUs.

Table 3. Different states of the classification algorithm starting from class centers (c_1, c_2, c_3) = (1, 30, 255).

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	1.00	30.00	255.00	1.47E+08
2	4.32	102.35	151.79	1.43E+08
3	24.68	100.37	146.29	1.42E+06
4	28.60	101.60	146.03	4.53E+04
5	28.60	101.60	146.03	0.00E+00

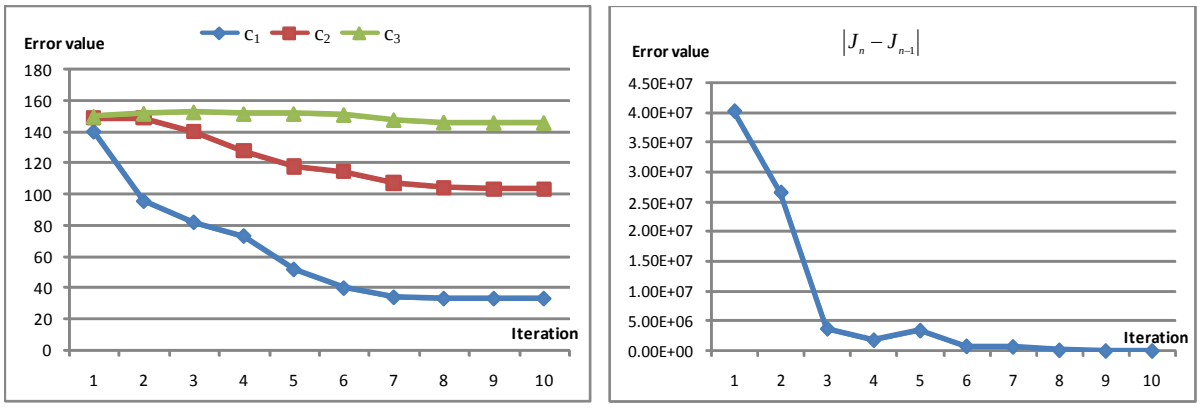


(a) (b)

Figure 8. Dynamic changes of the class centers starting from (c_1, c_2, c_3) = (1, 30, 255). (a) Class centers; (b) Error of the objective function.

Table 4. Different states of the classification algorithm starting from class centers (c_1, c_2, c_3) = (140, 149, 150).

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	140.00	149.00	150.00	4.01E+07
2	95.62	148.99	151.95	2.65E+07
3	82.13	140.26	152.83	3.72E+06
4	73.36	127.72	151.79	1.81E+06
5	51.98	117.99	151.79	3.41E+06
6	40.07	114.55	151.15	7.80E+05
7	34.22	107.44	147.88	6.90E+05
8	33.37	104.04	146.29	1.21E+05
9	33.37	103.50	146.03	2.45E+03
10	33.37	103.50	146.03	0.00E+00



(a) (b)

Figure 9. Dynamic changes of the class centers starting from (c_1, c_2, c_3) = (140, 149, 150). (a) Class centers; (b) Error of the objective function.

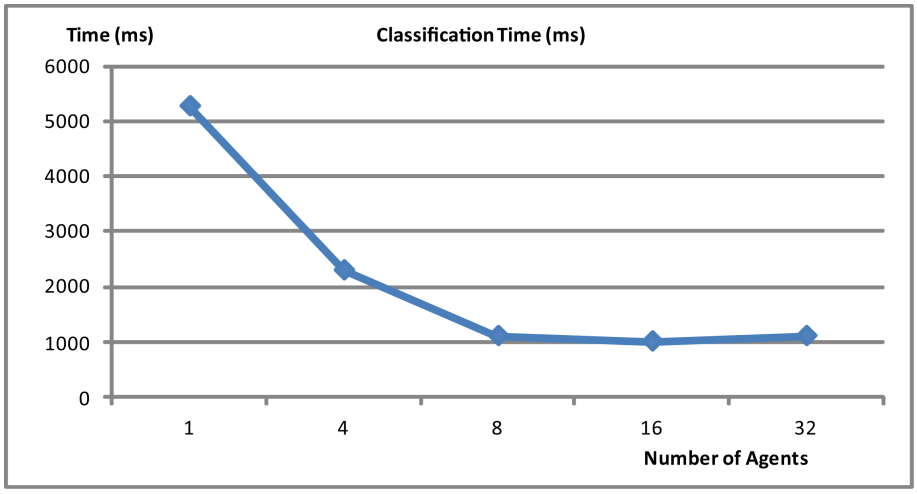


Figure 10. Time of classification (Tc) depending on the number of agents and with initial class centers (c_1, c_2, c_3) = (1, 2, 3).

5. Conclusion

In this paper, we have presented the distributed c-means classification algorithm and its implementation on a cooperative platform based on mobile agents. We considered a new method that allows us to distribute the c-means algorithm and encapsulate its program in mobile agents. In order to execute the parallel programs in our grid computing, the data are splitted and the program instructions are shared between the mobile agents. We considered a segmentation application algorithm on MRI cerebral image which is performed on this new cooperative multi-agent platform. The obtained results show that we can reduce the complexity of the parallel programs thanks to the benefits of mobile agents. Moreover, it contributes to overcoming the big data challenges and offers a high-performance computing tool using multi-agent system. This cooperative agent platform is flexible and extensible for parallel algorithms trends.

References

[1] Depuru, S.S.S.R., Wang, L.F., Devabhaktuni, V. and Green, R.C. (2013) High Performance Computing for Detection of Electricity Theft. *International Journal of Electrical Power & Energy Systems*, **47**, 21-30. http://dx.doi.org/10.1016/j.ijepes.2012.10.031

[2] Coria, J.A.G., Castellanos-Garzón, J.A. and Corchado, J.M. (2014) Intelligent Business Processes Composition Based on Multi-Agent Systems. *Expert Systems with Applications*, **41**, 1189-1205. http://dx.doi.org/10.1016/j.eswa.2013.08.003

[3] Sánchez, D., Isern, D., Rodríguez-Rozas, Á. and Moreno, A. (2011) Agent-Based Platform to Support the Execution of Parallel Tasks. *Expert Systems with Applications*, **38**, 6644-6656. http://dx.doi.org/10.1016/j.eswa.2010.11.073

[4] El-Rewini, H. and Abd-El-Barr, M. (2005) Advanced Computer Architecture and Parallel Processing. Wiley, Hoboken.

[5] Migliardi, M., Baglietto, P. and Maresca, M. (1998) Virtual Parallelism Allows Relaxing the Synchronization Constraints of SIMD Computing Paradigm. *High-Performance Computing and Networking Lecture Notes in Computer Science*, **1401**, 784-793. http://dx.doi.org/10.1007/BFb0037206

[6] Youssfi, M., Bouattane, O. and Bensalah, M.O. (2010) On the Object Modelling of the Massively Parallel Architecture Computers. *IASTED International Conference on Software Engineering*, Innsbruck, 71-78.

[7] Youssfi, M., Bouattane, O., Benchara, F.Z. and Bensalah, M.O. (2014) A Fast Middleware for Massively Parallel and Distributed Computing. *International Journal of Research in Computer and Communication Technology*, **3**, 429-435.

[8] Hwu, W.-M. (2014) What Is Ahead for Parallel Computing. *Journal of Parallel and Distributed Computing*, **74**, 2574-2581. http://dx.doi.org/10.1007/BFb0037206

[9] Rodríguez-González, A., Torres-Niño, J., Hernández-Chan, G., Jiménez-Domingo, E. and Alvarez-Rodríguez, J.M. (2012) Using Agents to Parallelize a Medical Reasoning System Based on Ontologies and Description Logics as an Application Case. *Expert Systems with Applications*, **39**, 13085-13092. http://dx.doi.org/10.1016/j.eswa.2012.05.093

[10] Benchara, F.Z., Youssfi, M., Bouattane, O., Ouajji, H. and Bakkoury, J. (2014) A Fast Middleware of Parallel and Distributed Virtual Machine Based on Mobile Agents for High Performance Computing. *Proceedings of CMT*14, Morocco, 260-263.

[11] Ni, L.M. and Jain, A.K. (1985) A VLSI Systolic Architecture for Pattern Clustering. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **PAMI-7**, 80-89. http://dx.doi.org/10.1109/TPAMI.1985.4767620

[12] Tsai, H.R. and Horng, S.J. (1999) Optimal Parallel Clustering Algorithms on a Reconfigurable Array of Processors with Wider Bus Networks. *Image and Vision Computing*, **17**, 925-936. http://dx.doi.org/10.1016/S0262-8856(98)00167-X

[13] Bouattane, O., Cherradi, B., Youssfi, M. and Bensalah, M.O. (2011) Parallel Cmeans Algorithm for Image Segmentation on a Reconfigurable Mesh Computer. *Parallel Computing*, **37**, 230-243. http://dx.doi.org/10.1016/j.parco.2011.03.001

[14] Padgham, L. and Winikoff, M. (2004) Developing Intelligent Agent Systems. Wiley, Hoboken. http://dx.doi.org/10.1002/0470861223