Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A NEW PARALLEL AND DISTRIBUTED FRAMEWORK BASED ON MOBILE AGENTS FOR HPC: SPMD APPLICATIONS

[1]FATÉMA ZAHRA BENCHARA, [2]MOHAMED YOUSSFI, [3]OMAR BOUATTANE, [4]HASSAN OUAJJI

[1]Laboratory SSDIA , ENSET Mohammedia, Hassan II University of Casablanca, Morocco
[2]Laboratory SSDIA , ENSET Mohammedia, Hassan II University of Casablanca, Morocco
[3]Laboratory SSDIA , ENSET Mohammedia, Hassan II University of Casablanca, Morocco
[4]Laboratory SSDIA , ENSET Mohammedia, Hassan II University of Casablanca, Morocco

E-mail: [1] benchara.fatemazahra@gmail.com, [2] med@youssfi.net, [3] o.bouattane@gmail.com, [4] ouajji@enset-media.ac.ma

ABSTRACT

This paper proposes a new distributed framework and its main components for HPC (High Performance Computing). It is based on a cooperative mobile agents model which implements the team works strategies to perform parallel programs execution as distributed one. The program and data to be performed is encapsulated on team leader agent which deploys its worker agents AVPUs (Agent Virtual Processing Units). All the AVPUs have to move to a specific node and perform and provide their computational results. Consider the great number of data and of the AVPUs to be managed by the team leader agent and which alter negatively the HPC. In this work we focused on introducing a specific mobile agent the MPA (Mobile Provider Agent) which implements some mechanisms for the management of data and tasks and the AVPUs to ensure a load balancing model. It applies also some additional strategies to maintain the others performance keys thanks to the mobile agents several skills.

Keywords: *High Performance Computing, Distributed Computing Environment, SPMD Applications, Mobile Agents, Big Data Processing*

1. INTRODUCTION

Nowadays, everyone need to get information, results and achieve tasks in real time. So it is possible by the use of the computer science technologies which make the complex tasks easy in order to perform these. For example running an application of weather predictions which is based on a big number of data and complex simulations using just one or two processors can be a hard task for the machine and sometime impossible to achieve the results. Consider these applications requirements. We need to introduce cooperation amongst processing power of different machines.

The Parallel Computing [1] is widely exploited to overcome these challenges with its flexible and extensible architectures such as: SIMD (Single Instruction multiple Data) and MIMD (Multiple Instruction Multiple Data); and topologies such as: 2D Mesh. Many fast parallel machines are developed in order to be flexible with the applications needs but they presented another challenges according to their high cost and to their limitation on the test and validation of new parallel algorithms. So the use of the PVM (Parallel Virtual Machine) [1] is considered as a suitable solution for these needs. This PVM machine is constituted over a grid computing using a set of heterogeneous machines connected with each other by the middleware. In [2], the authors proposed a virtual machine using mesh connected computer MCC which becomes mesh with multiple broadcast in [3] and polymorphic torus in [4] and a reconfigurable mesh computer RMC with integrated network for each processing element in [5],[6],[10],[11] and which are improved in [12] by assigning a set of distributed VPEs (Virtual Processing Elements) objects for each processing element in the grid; and recently by the use of GPUs and FPGAs in [7],[8], [9]. We can say that by introducing the concept of the grid and especially the middleware, the parallel computing are converged to the parallel and distributed one where the computing performance

Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

depends on the quality and the performance of the middleware. The question now is how to achieve the HPC. For the load balancing problem some algorithms have been designed for distributed systems [13],[14],[15],[16],[17]. To move the loads in a distributed system, the authors have used in [18] mobile agents which migrate loads from overloaded nodes to the lightly loaded ones and considering that all the grid nodes are homogeneous.

In this context, related to all these previous works we are focused on the use of the middleware which is based on the mobile agents. It is considered as a new grateful computer science technology which is used in [19] in order to propose a new model for automatic construction of business processes based on multi agent systems. And also in [20] to improve the management, the flexibility and the reusability of grid like parallel computing architecture; and the time efficiency of a medical reasoning system in [21]. So Thanks to the several interesting mobile agents skills, we design and implement a parallel and distributed environment composed by the middleware which assigns and orchestrates a set of mobile agents as AVPUs (Agent Virtual Processing Units) for each physical processor in heterogeneous parallel and distributed grid computing. It implements some interesting mechanisms for load balancing, fault tolerance, and to reduce the communication cost in order to have a control about all the parallel and distributed computing challenges and ensure the HPC. This paper is organized as follows:

- We will describe the proposed model for parallel and distributed computing, its main components which are: the mobile Team leader agents and the mobile Team worker agents and the MPA agent (Mobile Provider Agent) in the section 2.
- The section 3 is focused on presenting several mechanisms used by the Mobile Provider Agent in order to perform a load balancing middleware and a high performance parallel and distributed computing.
- Some interesting results performed by implementing the c-means and the fuzzy c-means algorithm in this model will be presented in section 4.

2. DISTRIBUTED COMPUTING ENVIRONMENT ARCHITECTURE

2.1 Distributed Computing Environment Model

Distributed Computing Environment is a new scalable and robustness model for performing distributed HPC of parallel programs as distributed one on a distributed system. It constitutes a parallel and distributed grid computing which is flexible with different topologies: 2D Mesh, 3D Mesh… and architectures: SIMD, SPMD (Single Program Multiple Data), MIMD, MPMD (Multiple Program Multiple Data)... It is based on a cooperative mobile agent team works as (AVPUs) deployed in each machine in order to perform parallel and distributed tasks. For example in Figure 1, in order to perform big data image segmentation, the fine grained c-means clustering algorithm is performed with SPMD architecture. It is implemented according to the distributed implementation in [26]. The Team leader (AVPU) divides the big data into elementary data and distributes them to their AVPUs. All the AVPUs perform in parallel the distributed image segmentation tasks and send the results to their Team leader agent to perform the big data segmented output images.

The distributed computing in this grid needs to be managed in order to ensure the HPC performance keys. This model has been extended by introducing the MPA agent which implements interesting mechanisms for managing the mobile

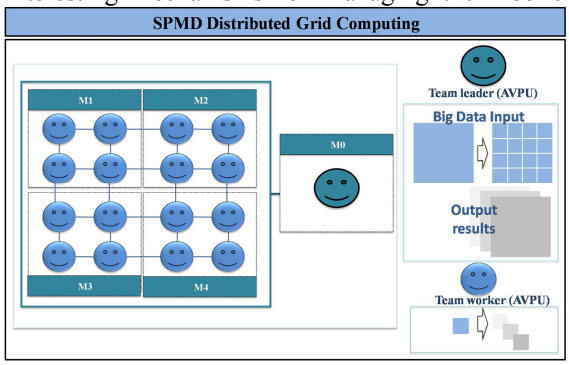


Figure 1: 2D Mesh Grid Computing for distributed image segmentation based on (AVPUs)

2.2 Model Main Components Overview

This distributed computing environment in Figure 2, is based on the power of the middleware and the mobile agents. The HPC of the parallel programs is performed in this environment by it cooperative main components which are created in these different environment states:

- Launching state: The middleware creates the Node Agent Container for each involved

Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

machine in the distributed computing and connects each of it in order to constitute the grid computing. We distinguish two particular nodes: the Node Host Agent Container created for the first machine responsible for launching this environment; and the Node Provider Agent Container where the MPA agent will be deployed.

- Deployment state: When the parallel program is deployed, the middleware deploys the MPA agent and the Team leader AVPU for each node which encapsulates tasks and creates their Team workers AVPUs. We can have one or two Team leader AVPU in the same node according to the number of the parallel programs deployed in this environment. Also each AVPU are autonomous and can decide to replicate itself in order to ensure a fault tolerance environment.

- Running state: When the parallel program is running, the team leader AVPU sends the tasks and data to the MPA agent in order to manage and provide them to the Team workers AVPUs by ensuring the load balancing of tasks execution in the grid computing. At the end, the MPA agent sends the results to the Team leader AVPU in order to perform the final results and return it to the MPA agent in order to be broadcasted for different nodes in the grid.

managing the pool of tasks and data and the results. It is an intelligent mediator between the Team leader AVPU and the Team workers AVPUs. This MPA agent has the knowledge of the number of team workers and its nodes performance. It manages a set of distributed pools by introducing the priority of the execution and the agent AID (Agent Identifier) for each tasks and data in these pools. So the team workers AVPUs can easily follow their data and tasks when they move to the MPA agent container. The MPA agent has also the ability to decide according to the parallel program architecture when to send the tasks and data and when to keep the agents to move to the pools. So by the implementation of the MPA agent in our model we have a control about the load balancing problem, and we reduce at the same time the communication cost. Also the MPA agent is autonomous, it can decide to move and to clone itself and resume its work in order to ensure a fault tolerance environment.

3.2 HPC Parallel and Distributed Computing Model

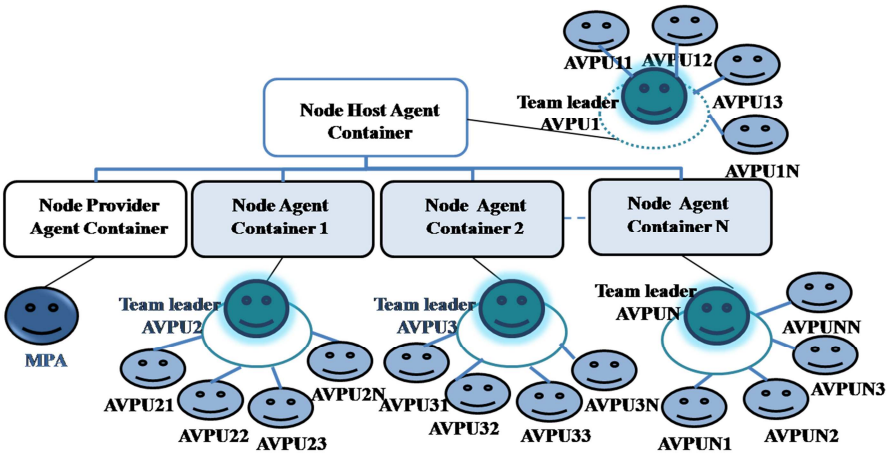


Figure 2: Distributed Computing Environment main components overview

3. FROM PARALLEL TO DISTRIBUTED COMPUTING

3.1 A Fast Distributed Computing Middleware

As this environment is constituted over heterogeneous machines with different degree of performance, we need some additional component in our model which is the MPA agent presented in Figure 3. The MPA agent is responsible for

This Model uses a specific mobile agent the MPA agent which implements interesting mechanisms for managing the cooperative mobile agent team work. It ensures the HPC performance keys (Communication cost, Fault Tolerance and Load Balancing) in Figure. 4 as follows:

Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

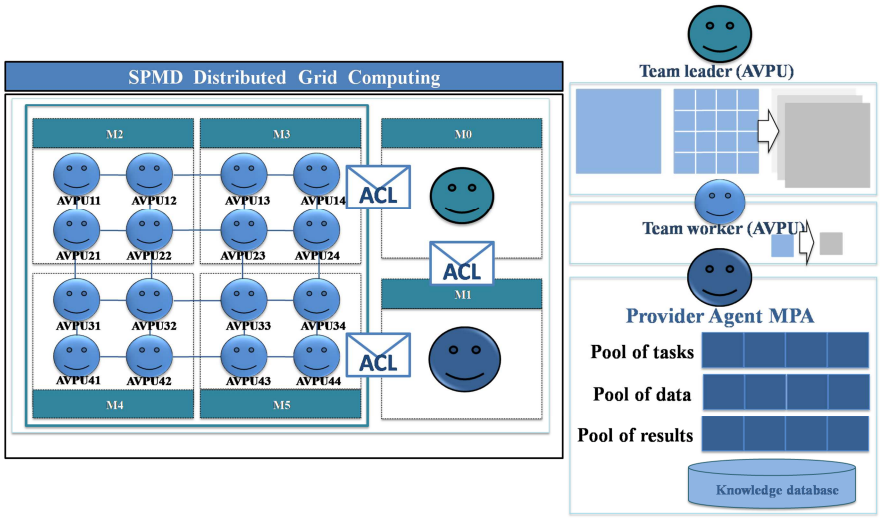


Figure 3: Distributed Middleware mechanisms for Computing Management

1) *Communication Cost*: The MPA agent use its asynchronous communication ability by exchanging ACL messages between the different components of the team work. This agent skill grants the ability to the team work for performing computation and communication at the same time. And also to the MPA agent for performing management of the team works and communication.

2) *Fault Tolerance:* The MPA Agent has the ability to clone itself at a specific time. So when some problems happen, the cloned MPAc agent starts. It resumes its state by the data and tasks of MPA agent and continues the tasks execution.

3) *Load Balancing:* The AMS (Agent Management System) agent performs the nodes performance monitoring and assembles and stores the results in the knowledge database. So the MPA agent accesses the data and performs its tasks by making some strategies to manage distribution of data and tasks to the team works in order to ensure load balancing computing.

3.3 Cooperative Mobile Agents Model Implementation

In Figure 5, we describe a scenario about the interaction between the different model main components in order to perform the execution of the parallel programs. This model is implemented using JADE (Java Agent Development Framework) [22].

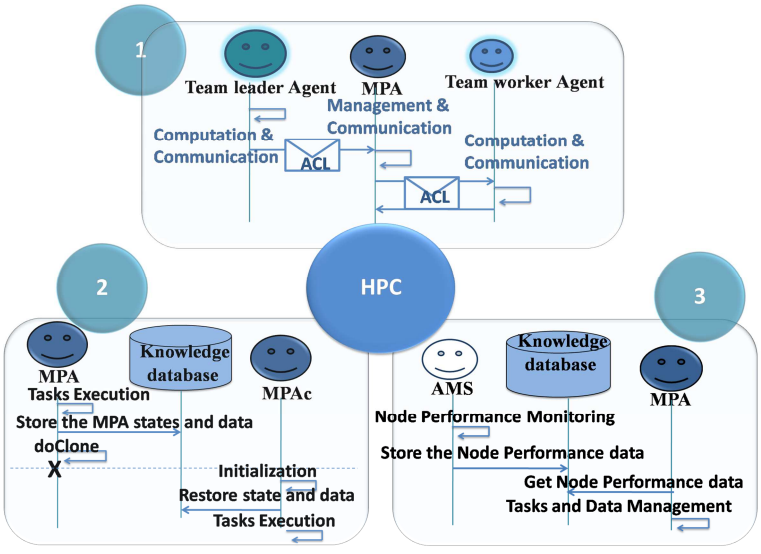


Figure 4: Overview of the Model management strategies for the main parallel computing challenges

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. SPMD APPLICATIONS

This section is investigated to demonstrate the performance of this model for SPMD applications. The two fined grained algorithms: Fuzzy c-means and c-means are implemented as distributed programs: DFCM (Distributed Fuzzy C-Means) and DCM (Distributed C-Means) and assigned to the Team leader Agent in order to perform medical image segmentation.

4.1 Distributed Implementation

The parallel c-means algorithm as defined in [23] and the FCM (Fuzzy C-Means) which is proposed by Dunn [24] and extended by Bezdek [25] are implemented in this environment as distributed programs in order to perform distributed image segmentation.

It is performed using the corresponding following steps presented under a sequence diagram in Figure 6. When the distributed program is assigned to the team leader agent which cooperates with its team works in order to perform the big data image segmentation.

4.2 Program Results

The scalability and the efficiency of this model are shown under the implementation of the both programs: c-means and Fuzzy c-means for distributed segmentation. Each program is assigned to a Team leader Agent which deploys its team workers and performs the segmentation for different input images: MRI cerebral image (Img1) in Figure 7 (a) and MRI cardiac image in Figure 8 (a) (Img2). And the Figures 7(b)-(c) and the Figures 8(b)-(c), are the c output segmented images for the both images respectively.

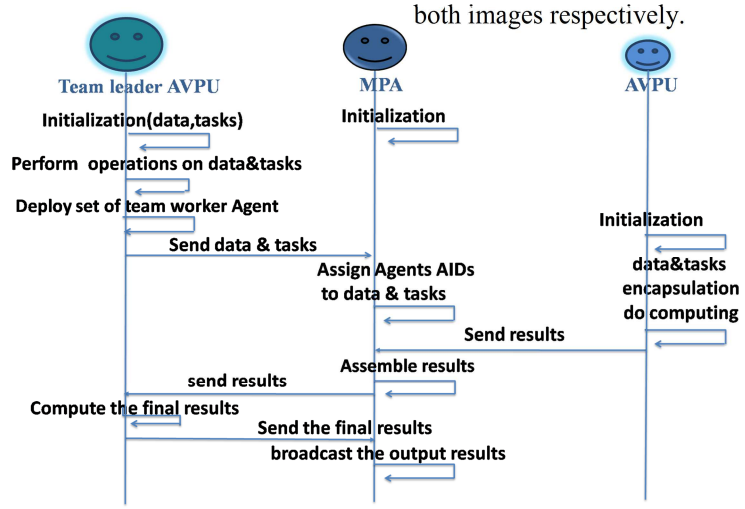


Figure 5: Sequence diagram for cooperative and distributed computing model

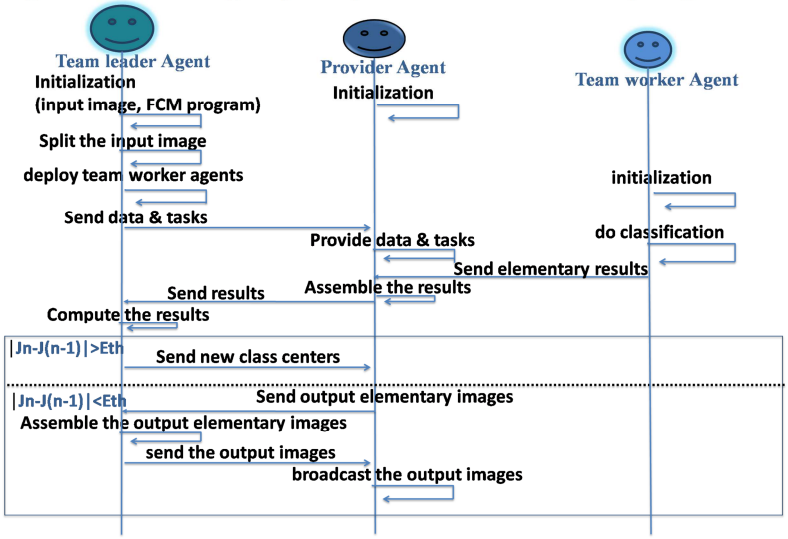


Figure 6: Sequence diagram for cooperative and distributed computing model

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The obtained results are investigated under these two following cases of dynamic convergence studies and the segmentation time analysis under the third case.

1) *Case 1:* For the same input image (Img1) and the initial class centers (c1, c2, c3, c4, c5) = (49.2, 50.8, 140.5, 240.5, 249.8). The DCM program converges in Table 1 to the final class centers after 15 iterations. And the DFCM program, in Table 2 converges to the final class centers after 6 iterations. The dynamic convergence for both programs is summarized in Figure 9.

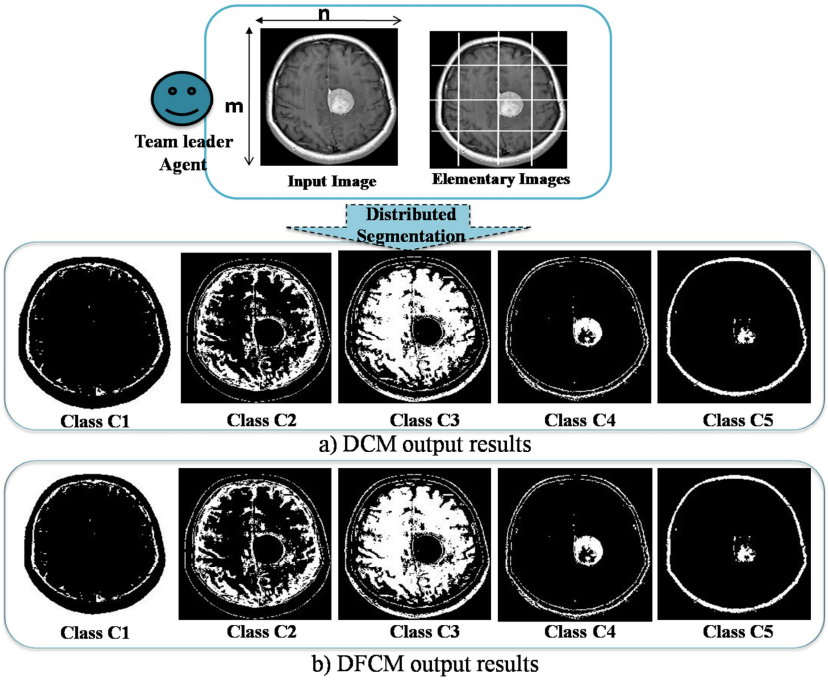


Figure 7: Output MRI cerebral images results by the distributed segmentation. a) The DCM program results; b) The DFCM program results

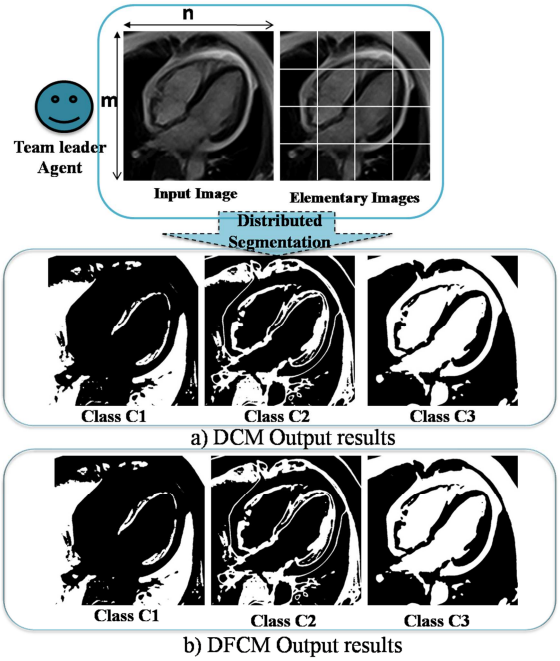


Figure 8: Output MRI cardiac images results by the distributed segmentation. a) The DCM program results; b) The DFCM program results

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2) *Case 2*: For the same input image (Img2) and the initial class centers (c1, c2, c3) = (1.5, 2.2, 3.8). The DCM program converges in Table 3 to the final class centers after 11 iterations. And the DFCM program, in Table 4 converges to the final class centers after 13 iterations. The dynamic convergence for both programs is summarized in Figure 10.

3) Case 3: We present the segmentation time analysis of the Img1 in Table 5 and of the Img2 in Table 6 using the DCM and DFCM program depending on the number of AVPUs involved in the segmentation. It is achieved over a distributed grid computing constituted over 8 heterogeneous CPUs. As illustrated in Figure 11 and Figure 12, the DCM algorithm is faster than the DFCM algorithm.

Table 1: Different states of DCM program for (Img1) segmentation starting from class centers (c1, c2, c3, c4, c5) = (49.2, 50.8, 140.5, 240.5,249.8).

Iteration	Value of each class center					Absolute value of the error
	C_1	C_2	C_3	C_4	C_5	$\|J_n-J_{n-1}\|$
1	49.2	50.8	140.5	240.5	249.8	1.34E+06
2	6.810	80.816	116.739	219.488	251.929	8.87E+05
3	4.586	80.899	114.175	201.473	249.149	5.04E+04
4	4.586	80.899	112.181	190.175	244.954	1.47E+04
5	4.586	79.486	110.073	184.334	242.084	7.51E+03
6	4.586	78.709	108.902	180.879	240.851	2.81E+03
7	4.043	76.425	107.215	178.037	239.649	6.67E+03
8	3.790	74.261	105.811	175.541	238.277	3.77E+03
9	3.790	73.316	105.063	173.497	237.385	7.29E+02
10	3.343	71.708	104.403	172.304	236.932	3.25E+03
11	3.082	71.343	104.323	171.858	236.932	2.21E+03
12	3.082	69.498	103.368	171.858	236.932	1.07E+03
13	2.876	68.258	102.866	171.382	236.932	1.70E+03
14	2.688	67.959	102.866	171.382	236.932	1.31E+03
15	2.688	67.959	102.866	171.382	236.932	0.00E+00

Table 2: Different states of DFCM program for (Img1) segmentation starting from class centers (c1, c2, c3, c4, c5) = (49.2, 50.8, 140.5, 240.5, 249.8).

Iteration	Value of each class center					Absolute value of the error
	C_1	C_2	C_3	C_4	C_5	$\|J_n-J_{n-1}\|$
1	49.2	50.8	140.5	240.5	249.8	5.67E+06
2	37.901	44.168	121.629	222.386	243.506	6.45E+04
3	18.423	38.42	108.49	207.129	244.039	9.39E+04
4	3.403	46.439	102.188	195.174	244.515	1.27E+05
5	1.132	53.31	100.327	186.73	243.734	3.36E+04
6	1.175	56.941	100.092	181.077	242.571	4.82E+03

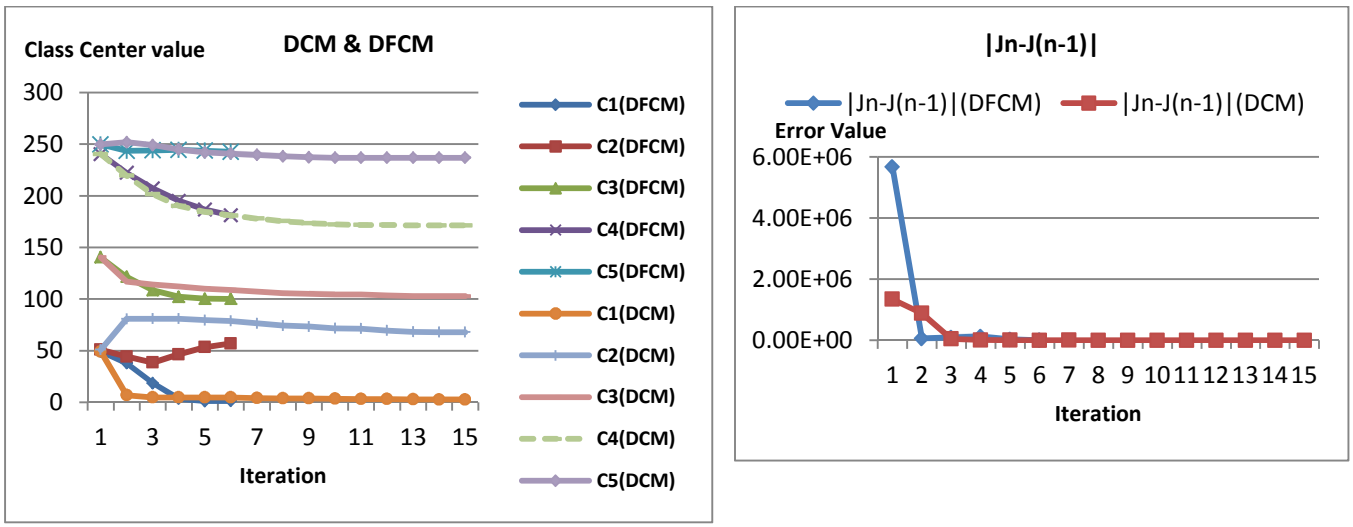


Figure 9: Dynamic changes of the class centers starting from centers (c1,c2,c3,c4,c5)=(49.2,50.8,140.5,240.5,249.8) for DCM and DFCM program using (img1). (a) Class centers; (b) Error of the objective function

Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

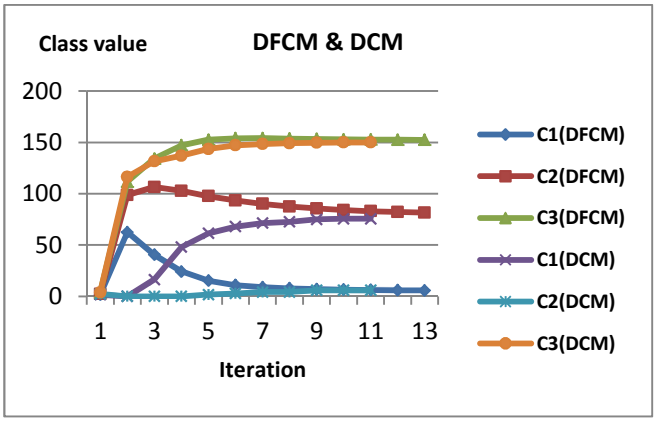
However, by the use of the DFCM method we grant a good precision of the class centers and of the quality of the MRI image segmentation compared to the DCM method. And we notice that the segmentation time of the two methods is reduced according to the number of AVPUs. Thus, demonstrates the speedup of the distributed segmentation compared to the sequential one. For example for the DFCM method using 32 AVPUs, we perform interesting speedup of 6 for (Img1) and of 8 for (Img2) in Figure 11(b) and in Figure 12(b). And we can see clearly that from 16 AVPUs, the speedup achieves its maximum value for the two algorithms. This is due to the size of the elementary images which become very small; and enhance the latency and influence the performance of distributed systems. In this case it is interesting to perform the segmentation of these images by the Team leader Agent instead of deploying a set of AVPUs.

Table 3: Different states of DCM program for (Img2) segmentation starting from class centers (c1, c2, c3)=(1.5, 2.2, 3.8).

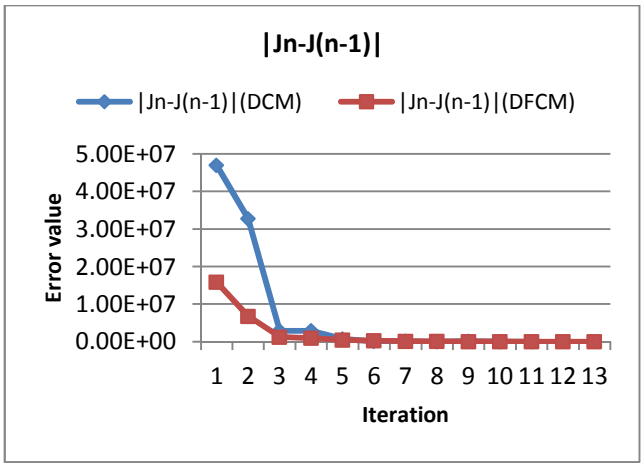
Iteration	Value of each class center			Absolute value of the error
	C_1	C2	C3	$\|J_n-J_{n-1}\|$
1	1.5	2.2	3.8	4.70E+07
2	0.0	0.0	116.342	3.28E+07
3	16.343	0.0	131.467	2.90E+06
4	48.002	0.0	136.958	2.89E+06
5	61.311	1.651	143.605	7.42E+05
6	67.935	2.734	147.181	7.67E+04
7	71.309	4.149	148.424	9.15E+04
8	72.599	4.149	149.206	3.98E+03
9	74.951	5.886	149.585	1.21E+05
10	75.553	5.886	149.950	9.15E+03
11	75.553	5.886	149.950	0.00E+00

Table 4: Different states of DFCM program for (Img2) segmentation starting from class centers (c1, c2, c3)=(1.5, 2.2, 3.8).

Iteration	Value of each class center			Absolute value of the error
	C_1	C2	C3	$\|J_n-J_{n-1}\|$
1	1.5	2.2	3.8	1.59E+07
2	62.517	98.681	111.686	6.76E+06
3	40.436	106.484	134.142	1.26E+06
4	24.141	102.733	147.129	9.25E+05
5	15.173	97.560	152.552	4.80E+05
6	10.864	93.349	154.062	2.76E+05
7	8.811	90.014	154.100	1.23E+05
8	7.692	87.430	153.728	7.10E+04
9	6.988	85.466	153.319	5.23E+04
10	6.507	83.996	152.975	3.23E+04
11	6.168	82.909	152.710	2.16E+04
12	5.928	82.114	152.514	1.83E+04
13	5.757	81.537	152.371	1.15E+04



a)



b)

Figure 10: Dynamic changes of the class centers starting from centers (c1,c2,c3)=(1.5,2.2,3.8) for DCM and DFCM program for (img1) segmentation. (a) Class centers; (b) Error of the objective function

Table 5: Segmentation time of DCM and DFCM programs for (Img1) according to AVPUs number.

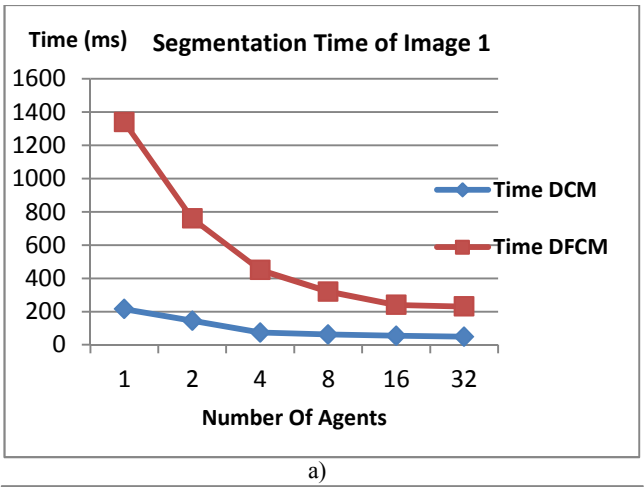
Number of Agents	DCM	DFCM
1	215	1339
2	145	760
4	73	450
8	62	320
16	54	240
32	48	230

Table 6: Segmentation time of DCM and DFCM programs for (Img2) according to AVPUs number.

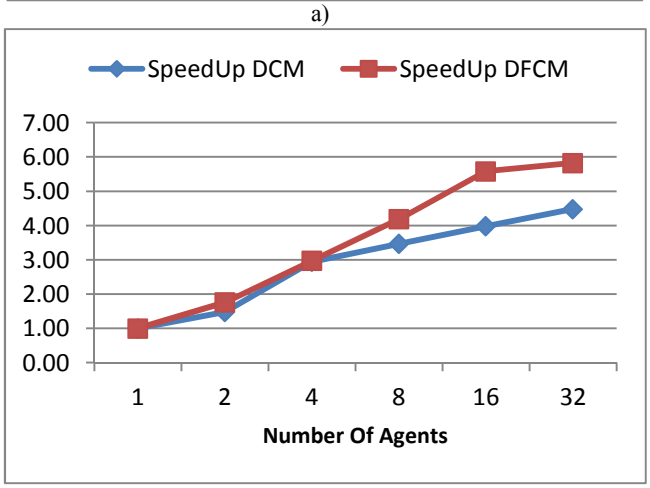
Number of Agents	DCM	DFCM
1	554	12475
2	320	5340
4	250	3123
8	130	1780
16	123	1650
32	123	1560

Journal of Theoretical and Applied Information Technology
20th October 2015. Vol.80. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

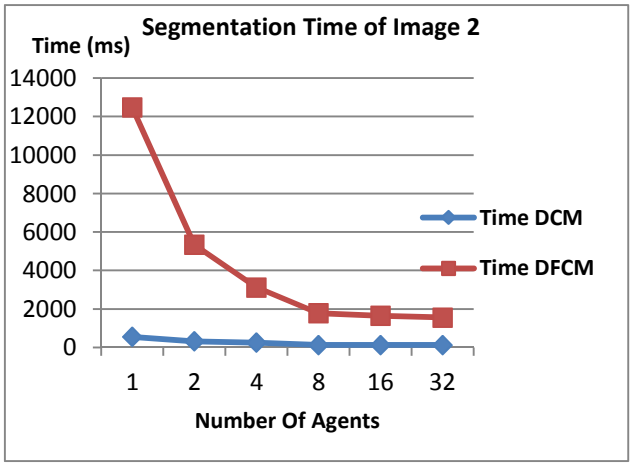


a)

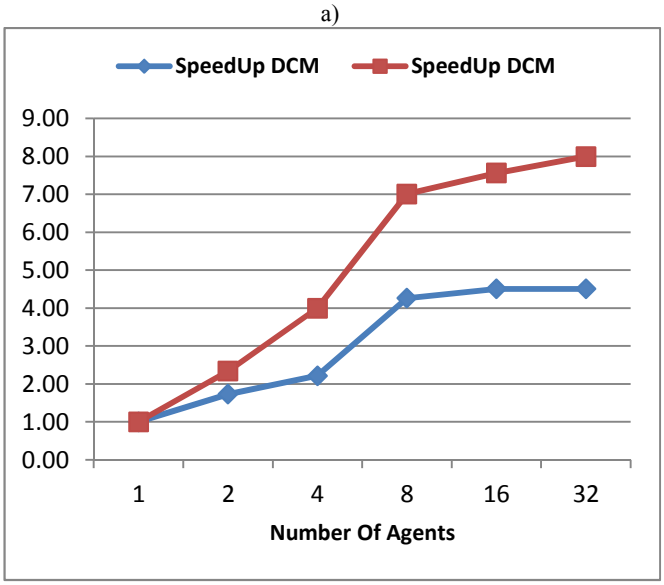


b)

Figure 11: Segmentation time (Time DCM) and (Time DFCM) of the (Img1) using the initial class centers (c1,c2,c3,c4,c5)=(49.2,50.8,140.5,240.5,249.8) and DCM and DFCM program; (a) Segmentation times depending on the number of AVPUs ; (b) The gain of DCM and DFCM methods



a)



b)

Figure 12: Segmentation time (Time DCM) and (Time DFCM) of the (Img2) using the initial class centers (c1,c2,c3)=(1.5,2.2,3.8); (a) Segmentation times depending on the number of AVPUs ; (b) The gain of DCM and DFCM methods

5. CONCLUSION

In this paper, we have presented a distributed framework for high performance parallel and distributed computing and its applications for two fine grained clustering algorithms: c-means and the Fuzzy c-means algorithms. It is based on a new middleware which is flexible with parallel programs trends and which use the Mobile Agents as Agent Virtual Processing Units AVPUs. Each mobile agent associates its skills: autonomy, and mobility, and communication ability using ACL messages to provide the computing environment

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that ensure the performance keys (load balancing, fault tolerance and reduce the communication cost). Moreover, its implements the Mobile Provider Agent (MPA) which use interesting mechanisms for ensuring this performance keys and improve the HPC. The scalability and the efficiency of this framework are demonstrated by the segmentation time and the speedup results performed for these distributed algorithms compared to the sequential one. We anticipate implementing some management strategies for input data and AVPUs managements in order to reduce the latency in this environment. And lead to HPC enhancement for SPMD Applications and for different parallel computing architectures trends.

REFRENCES:

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

[2] R. Miller, et al., "Geometric algorithms for digitized pic-tures on a mesh connected computer", IEEE Transactions on PAMI, Vol.7 ,1985, pp.216-228.

[3] V.K. Prasanna, D.I. Reisis, "Image computation on meshes with multiple broadcast", IEEE Transactions on PAMI, Vol.11, 1989, pp.1194-1201.

[4] H. LI, M. Maresca, "Polymorphic torus network", IEEE Transaction on Computer 1989, C-38, pp.1345-1351.

[5] T. Hayachi, K. Nakano, S. Olariu, "An O ((log log n)2) time algorithm to compute the convex hull of sorted points on re-configurable meshes", IEEE Transactions on Parallel and Distributed Systems , Vol. 9, 1998, pp.1167-1179.

[6] R. Miller, V.K. Prasanna-Kummar, D.I. Reisis, Q.F. Stout, "Parallel computation on re-configurable meshes", IEEE Transactions on Computer, Vol. 42, 1993, pp.678-692.

[7] Y. Zhao, F.C.M. Lau, "Implementation of Decoders for LDPC Block Codes and LDPC Convolutional Codes Based on GPUs", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, 2014, pp.663-672.

[8] J. Wu, J. JaJa, E. Balaras, "An Optimized FFT-Based Direct Poisson Solver on CUDA GPUs", IEEE Transactions on Parallel and Distributed Systems 2014, Vol. 25, pp.550-559.

[9] A. Rafique, G.A. Constantinides, N. Kapre, "Communication Optimization of Iterative Sparse Matrix-Vector Multiply on GPUs and FPGAs, IEEE Transactions on Parallel and Distributed Systems 2014.

[10] M. Youssfi, O. Bouattane, M.O. Bensalah, "A Massively Parallel Re-Configurable Mesh Computer Emulator: Design, Modeling and Realization", J. Software Engineering & Applications, Vol. 3, 2010, pp.11-26.

[11] O. Bouattane, B. Cherradi, M.Youssfi, M.O, Bensalah, M.O., "Parallel c- means algorithm for image segmentation on a reconfigurable mesh computer", ELSEVIER, Parallel computing, Vol. 37, 2011, pp.230-243.

[12] M.Youssfi, O. Bouattane, F.Z. Benchara, M.O. Bensalah, "A Fast Middleware For Massively Parallel And Distributed Computing", Inter. Journal of Research in computer and Communication Technology, Vol. 3, 2014,pp. 429-435.

[13] D.R. Karger, M. Ruhl, "Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems", Theory of Computing Systems, Vol. 39, 2006, pp.787-804.

[14] I. Konstantinou, D. Tsoumakos, N. Koziris, "Fast and Cost-Effective Online Load-Balancing in Distributed Range- Queriable Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 22, 2011, pp.1350-1364.

[15] H.C. Hsiao, H.Y. Chung, H. Shen, Y.C. Chao, "Load Rebalancing for Distributed File Systems in Clouds", IEEE Transactions on Parallel and Distributed Systems, Vol. 24, 2013, pp.951-962.

[16] H.C. Hsiao, C.W. Chang, "A Symmetric Load Balancing Algorithm with Performance Guarantees for Distributed Hash Tables", IEEE Trans. Computers, Vol. 62, 2013, pp.662-675.

[17] G. Cybenko, "Load balancing for distributed memory multiprocessors", Journal of Parallel and Distributed Computing, Vol. 7, 1989, pp. 279-301.

[18] J. Liu, X. Jin, Y. Wang, "Agent-Based Load Balancing on Homogeneous Mini-grids: Macroscopic Modeling and Characterization", IEEE Transactions on Parallel and Distributed Systems, 2005, pp. 586-594.

[19] J.A. García Coria, J.A. Castellanos-Garzón, J.M. Corchado, "Intelligent business processes composition based on multi-agent systems", ELSEVIER. Expert Systems with Applications Vol. 41, 2014, pp.1189-1205.

[20] D.Sánchez, D. Isern, A. Rodríguez-Rozas, Moreno, "Agent-based platform to support the execution of parallel tasks", ELSEVIER, Expert

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Systems with Applications, Vol. 38, 2011, pp. 6644-6656.

[21] A. Rodríguez-González, J. Torres-Niño, G. Hernández-Chan, E. Jiménez-Domingo, J.M. Alvarez-Rodríguez, "Using agents to parallelize a medical reasoning system based on ontologies and description logics as an application case", ELSEVIER, Expert Systems with Applications Vol. 39, 2012, pp.13085-13092.

[22] F.L. Bellifemine, G. Caire, D. Greenwood, "Developing Multi-Agent Systems with JADE" Wiley 2007.

[23] O. Bouattane, B. Cherradi, M. Youssfi, M.O. Bensalah, "Parallel cmeans algorithm for image segmentation on a reconfigurable mesh computer", ELSEVIER, Parallel computing, Vol. 37, 2011, pp.230-243.

[24] J.C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters", Journal of Cybernetics, Vol. 3, 1973, pp. 32-57.

[25] J.C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms", Plenum Press. New York 1981.

[26] FZ. Benchara, M. Youssfi, O. Bouattane, H. Ouajji, M.O. Bensalah, "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, Vol. 8, 2015, pp.103-113.