Performance Evaluation of Parallel Computing on Agent-Based Models: The Minority Game Case

Santiago Montiel, Sebastian Guala
Instituto de Ciencias, Universidad Nacional de General Sarmiento, Buenos Aires, Argentina
{smontiel, sguala}@ungs.edu.ar

Abstract: Simulations of agent-based models developed for topics of learning and inductive reasoning in artificial intelligence, social behavior, decision making, etc., are progressively requiring higher power processes while they increase their participation as management and political decisions support. In this work we develop the implementation of the Minority Game Model for HPC platforms in order to analyze the performance of simulations related to contexts of agent-based models for large scales. We compare times to parallel and sequential processes for several instances and get the corresponding speedup. For this work we use the MPI system with a hardware configuration of Master-Worker (Slave) paradigm with a cluster of upto 10 processors as workers. In order to improve efficiency, we evaluate performances for several sizes of clusters varying the size of the instances of the problem and detect optimum configurations for some instances of simulation.

Keywords: HPC, Minority Game, Simulations of Complex Systems.

1. Introduction

During the last decades, research in computer modeling and simulations of complex systems has become dominant. Scientific community has turned its attention to problems focused on the emergence of aggregates like economic behavior, social options, strategical choices, patterns of decision, etc., as a result of the dynamical interaction of many individuals with complex and heterogeneous behaviors within realistic environments [1]. These agent-based models (ABM) consider explicitly the dynamical approach by inductive (rather than deductive) evolution, adaptation and learning of individuals; through the study of many interacting degrees of freedom. ABM uses a bottom-up approach to analyze the complexity of human interactions and understand the emergence of global processes from individuals to the system as a whole. The key point in that kind of models is that the individual behavior of agents depends on the environment, the social interaction and the individual outcomes of previous actions. In many of these situations, the computational approach to dynamical interaction results more effective than other techniques like formal mathematical models.

After that description, it is clear that advances in ABM require increasingly high computational power. Despite lots of characteristics emergent behaviors can be observed in different scales, ABM simulations tend to be small and the agents definition tends to be simple. Therefore, researchers could be missing phenomena only observed in large-scale or in more complex scenarios. As those ABM become larger, they require much greater computing capability. In this sense, high-performance computing (HPC) systems allow to perform simulations at scales that can address the interactions of several millions of individuals with artificial intelligence algorithms and high computing costs. Therefore, new efforts involve the development of techniques and experiments focused in the application of HPC platforms to the new challenging scales of ABM [2]. In our case, by HPC platforms we focus on parallel programming over multiple processes with Message Passing Interface (MPI) applications.

In the context of ABM, one of the most popular models dealing with complex systems, inductive learning and social behavior which attracted the attention of the statistical mechanics community in the last decade is the so-called *Minority Game* problem. The Minority Game (MG) is a discrete adaptive model based on the "El Farol" bar problem [3], which was introduced in [4] in order to study strategical behaviors. It is associated to the analysis of simple financial markets due to the similar kind of binary decisions that must be made (e.g., "buy" or "sell") and to the bounded rationality and incomplete information that the individuals have. Given the number of scenarios where the MG-like applications can be encountered and the large amount of agents potentially involved, in this work we analyze the advantage of implementing HPC (in particular, MPI) platforms in order to improve the performance of models related to MG-like contexts. The rest of this paper is organized as follow. Section 2 defines formally the model. Section 3 shows the parallelization and the design of internodes communications. Section 4 shows experimental results. Finally, Section 5 gives conclusions and future works.

2. The Minority Game Model

The MG is an ABM inspired in real complex systems which presents interesting collective properties like coordination among agents [5]. However, from a more general sense, its behavior tries to capture essential characteristics of some real situations in which belonging to the minority group turns out to be the most convenient position (e.g., financial systems, traffic problems, data networks, fishing places, price promotions, etc.)[6][7][8][9][10][11]

In the MG model, *N* agents must independently choose between two options (usually denoted by 0 and 1), and the agents who make the minority decision win. Each agent's choice depends on a set of *s* strategies. A *strategy* is a prediction function.

Table 1: Example of typical strategy that tries to predict the minority side considering the last 3 previous outcomes.

m=3	Prediction
000	1
001	1
010	0
011	1
100	0
101	0
110	0
111	1

In our case, every agent has *s*=2 strategies at hand and at each step of the game, she plays using her best one (i.e., the one which most often has predicted the minority option).

To this end, the strategies that have predicted the correct outcome for each step are rewarded with one point, regardless of usage. Then, the strategies of every agent are ranked according to the number of rounds that each one has correctly predicted the minority side. Whenever an agent's two strategies are equally ranked, she chooses one of them randomly. Each strategy predicts the next winning option (0 or 1) by processing the sequence of outcomes from the last *m* time steps, which is the only available public information. The value *m* is known as the agents *memory*. Since every strategy contains the 2^m possible historic states (see Table 1), the whole pool has 2^{2^m} strategies; and at the beginning of the game each agent randomly draws her set of *s* of them (maybe repeated by chance). Far away from the possibility of knowing her maximization-of-payoff choice, each agent is confined to play in the way her own best performing strategy (i.e., the one from her set with the best score until that moment) suggests. The most studied variable in the MG is the reduced variance $\frac{\sigma^2}{N} = \langle \frac{\left(N_1 - \frac{N}{2}\right)^2}{N} \rangle$ [12]. It measures the population's waste of resources by averaging the quadratic deviation from *N*/2. When crowds emerge in the game, their contribution to $\frac{\sigma^2}{N}$ is very important, indicating that the minority is small, thus fewer resources are being allocated to the population as a whole. On the other hand, the main point of the game is that for certain values of the parameters *m*, *N*, and *s*, $\frac{\sigma^2}{N}$ results notably smaller than that obtained for a game in which each of the *N* agents randomly chooses between the two options. That is the most remarkable emerging behavior of the MG because it means that hundreds or thousands of agents, only based on a small public information and searching for self-benefit, are able to inductively learn to coordinate decisions to be on average in the minority more times than choosing side randomly. In particular, the region of maximum coordination is around $\frac{\sigma^2}{N} \approx 0.032$, thus, it is where our particular attention is paid.

Table 2: Example of a matrix of strategies for the sequential implementation of the MG for *m*=3 and *N* agents. The tuples (st x,ag y) mean *strategy x* of the agent *y*.

m=3	(st 1, ag 1)	(st 2, ag 1)	(st 1, ag 2)	(st 1, ag 1)	…	(st 2, ag N)
000	1	0	0	0	…	1
001	1	1	0	0	…	0
010	0	1	1	0	…	1
011	1	1	0	1	…	1
100	0	1	1	1	…	0

101	0	0	0	0	…	1
110	0	0	1	1	…	0
111	1	0	0	1	…	0
score	10	5	7	12	…	9

3. Implementation

For the sequential implementation of the MG, a matrix is constructed in such a way that every agent has her 2 strategies (see Table 2): each row of the first column corresponds to a history (i.e., the possible previous m outcomes). From the second column on: in the first two columns, two strategies are allocated corresponding to the agent 1; in the second two columns, two strategies are allocated corresponding to the agent 2, and so on. The last row of each column saves the score of the strategy, which is what the agent looks to decide one of her two strategies to follow to make the next choice. Initially, all the points are 0 and in case of tied scores, agent takes one of the couple randomly.

After getting the less chosen option, those strategies that guest the outcome are rewarded with one point and the history is updated.

The application of this model is parallelized by an implementation on a network of CPU's family [13] with processors Double - Intel Dual Core Xeon 5030 2.67 Ghz, memory Double - Kingston 1Gb 533Mhz DDR2 ECC FBDIMM, in which the middleware used is MPI under the hierarchical Master-Worker(Slave) paradigm [14][15] with up to 10 processors as workers and INFINIBAND HCA Cards MHEA28-2TC PCI-E. During the evaluation, we set different instances of MG and several random benchmarks are generated to test and compare performances.

As the parallel implementation is based on the Master-Worker in “star” configuration. Each worker has a proportional part of the whole matrix previously mentioned, corresponding to a subset of agents. Workers initialize their strategies (2 columns per agent and the last row save the score). Each worker makes their agents vote (agents bet according to the row corresponding to the given history and their best scored strategies), then compute their decisions and pass the sum to the Master. The Master receives the partial sums from the workers, obtains the minority option, updates the current history (which is the same for all the agents) and it is passed (in decimal system) to the workers by collective communication.

Table 3: Speedup for 1000 agents with m=10 and 100000 rounds and averaged over 10 realizations. The sequential time for this instance is 27.29 seconds.

Workers	MPI time	Speedup
2	6.69	4.08
4	6.13	4.45
5	5.99	4.56
8	6.91	3.95
10	7.64	3.57

Table 4: Speedup for 100000 rounds of 1000 agents with 5 workers and averaged over 10 realizations.

Memory m	Sequential Time	MPI time	Speedup
5	13.76	5.85	2.35
6	13.61	5.82	2.34
7	14.03	5.72	2.45
8	17.75	5.71	3.11
9	24.17	5.75	4.20
10	27.29	5.99	4.56

Finally, workers update the scores of their strategies and a new step begins. For instance, if the number of agents is 100 and the total sum is 54, it means the minority option is 0. For the parallel implementation, the data structure is a matrix of size 2*(number of agents / available workers).

4. Results

In order to evaluate acceleration of processes, different length of history were tested, which change the size of the matrices. In particular between m=5 (strategies have 129 rows, 128 histories + 1 for score) and m=10 (strategies have 1025 rows, 1024 histories + 1 for score). To analyze the performance for large scales of the model we set different instances of configuration from which some observations can be made. As it can be seen in Table 3, the speedup grows until the number of workers reaches 5. Then, it decays because communication process starts to play a significant role just where computation capacity for the size of such an instance leads the parallelization advantage [16][17] [18][19]. Therefore, with focus on 5 workers, note from Table that the speedup grows even more when the scale of the model is larger. As one can expect, we will see that such a 5-worker optimum is not longer effective for larger instances.

Table 5: Speedup for 1000000 rounds of 10000 agents with 5 workers (5W), 8 workers (8W) and 10 workers (10W), averaged over 10 realizations.

Memory m	Sequential Time	MPI time (5 w)	Speedup (5 w)	MPI time (8 w)	Speedup (8 w)	MPI time (10 w)	Speedup (10 w)
5	4600.09	189.59	24.26	150.74	30.62	143.22	32.12
6	4903.12	182.33	26.89	146.38	33.50	132.85	36.91
7	5132.12	176.49	29.08	142.57	36.00	134.40	38.18
8	5287.42	172.07	30.73	140.54	37.62	135.08	39.14
9	5421.53	168.91	32.10	138.01	39.28	130.98	41.39
10	5422.24	166.32	32.60	135.86	39.91	129.16	41.98

In order to study performances for large-scale instances, we fix, in turns, the number of workers in 5, 8 and 10 with 1000000 rounds of 10000 agents, varying the memory m between 5 and 10, as shown in Table . Consistently with what we saw previously, the larger both instances and scale of the model, the larger is the performance gap between parallel and sequential implementation. There we see that the speedup increases from 24.26x until 32.60x with 5 workers for m=5 and m=10, respectively; and from 32.12x until 41.98x with 10 workers for m=5 and m=10, respectively. Note that, as we mentioned above, we are focusing our particular attention to the region of maximum coordination of the MG model ($\frac{\sigma^2}{N} \approx 0.032$). Therefore, the reason why we do not go further m=10 is that between N=1000 and N=10000 the most coordinated behavior among agents appears around m=5 and m=10 [5]. This is the most important region, where the phase transition of the model takes place [12]. Therefore, even we see that in Table 5 the speedup grows as m increases, for instances of 10000 agents the simulations are lacking in interest further about m=10. As an interesting observation, note from Table that, as the size of memory grows, the speedup and the sequential time also grows, which is expected, but the MPI time decreases. Due to the size of the problem duplicates when the memory increases a unit, the reason of that is not completely clear as long as we know. However, several complementary testing simulations seem to confirm that. As one of these examples, note from Table that small instances of 100000 rounds and 1000 agents take a minimum around m=8. In principle, it could suggest that, for a given number of agents, the minimum MPI time might depend on the memory size. In fact, Table extends this observation to memory m=11, m=12, m=14 and m=16 for 5 workers and 1000000 rounds, to improve accuracy, and minimum MPI time is found around m=7 for 100 agents, m=9 for 1000 agents and m=12 for 10000 agents, then time increases in all cases.

Table 6: MPI time for 1000000 rounds with 5 workers and averaged over 10 realizations.

Memory m	100 agents	1000 agents	10000 agents
5	45.61	58.34	182.13
6	45.47	57.25	174.13
7	45.35	56.52	169.10
8	45.45	55.68	165.05
9	45.59	55.35	162.36
10	45.59	55.63	158.80
11	45.66	55.68	156.53

5. Conclusions

In this work we developed the implementation of the Minority Game model by implementing HPC (in particular, MPI) platforms in order to analyze the performance improvement of contexts related to agent-based models, specially for large scales. Our simulations took up to 11 nodes, used 10 processors of them as workers, covering performance from 2 to 10 workers. Measuring times for the parallel MPI implementation it was possible to see a maximum performance for 5 workers, given improvements of 4.85x, and a subsequent performance reduction when the number of 5 workers is exceeded for instances of 100000 rounds of 1000 agents with m=10. The reason was attributed to the communications capacity, which is known to start playing a significant role.

As usual instances of the MG typically concern scales of the order of 100000-decision rounds per run, with 1000 agents, in this work we set that size as a sort of lower bound and we evaluated larger instances until 1000000 rounds with 10000 agents. As an example in such a mentioned instance, MPI time was about 2 minutes while the sequential realizations took of the order of one and a half hour time. For memory sizes between 5 and 10 and 1000 agents, we have got the best performances with 10 workers (as long as we could try), with improvements of speedup of the order of 42x.

To what large scales of ABM concerns, it is interesting to note that MPI time keeps nearly constant for instances of sizes between 32 (m=5) and 1024 (m=10), whereas the sequential time grows. What we found from the time consuming point of view is that a heavy work is concentrated on the construction, decision and updating processes of the strategies matrix as the one seen in Table . Therefore, the HPC implementation for large scale ABM appears to be remarkably useful as the complexity of the model grows, not only to run the simulations but also for the initial startup.

Finally, it was observed that the MPI time decreases as memory size grows, and for some instances a minimum MPI time was found. As it would imply optimum memory sizes depends on the instance and the workers, further researches are strongly encouraged in order to explain the reasons, which can contribute to characterize ABM instances to improve future performances.

References

1. Helbing, D.: *Social Self-Organization: Agent-Based Simulations and Experiments to Study Emergent Social Behavior, Understanding Complex Systems Series.* Springer-Verlag (2012).

2. N. Collier : *Parallel agent-based simulation with Repast for High Performance Computing SIMULATION* (2012)

3. Arthur, B. : *Inductive Reasoning and Bounded Rationality (The El Farol Problem),.* American Economic Review, (A.E.A. Papers and Proc.) (1994)

4. D. Challet: *Emergence of cooperation and organization in an evolutionary game.* Physica A. (1997)

5. D. Challet: *Minority Games.* Oxford University Press (2005)

6. D. Guo: *The Modeling and Simulation of the Traffic Congestion Drift Induction Mechanism Based on Minority Game, 2nd International Conference on Networking and Information Technology IPCSIT vol.17 IACSIT Press* (2011).

7. G. Bianconi: *Effects of Tobin taxes in minority game markets, Journal of Economic Behavior and Organization* (2009)

8. K. Kim: *Dynamics of the minority game for patients.* Physica A (2004)

9. P. Mähönen: *Minority game for cognitive radios: Cooperating without cooperation, Physical Communication* (2008)

10. R.D. Groot: *Minority Game of price promotions in fast moving consumer goods markets.* Physica A. (2005)

11. T. Takama: *Forecasting the effects of road user charge by stochastic agent-based modelling,.* Transportation Research Part A (2008)

12. R. Savit: *Adaptive competition, market efficiency, and phasse transitions.* Phys. Rev. Lett. (1999)

13. *Centre for High Performance Computing.* (n.d.). Buenos Aires University, http://cecar.fcen.uba.ar.

14. G. Hager: *Introduction to High Performance Computing for Scientists and Engineers, Chapman and Hall/CRC Computational Science* (2010)

15. P. Pacheco.: *Parallel Programming with MPI, .* Morgan Kaufmann Publishers (1996).

16. A.I. El-Nashar: *To parallelize or not to parallelize, speed up issue, International Journal of Distributed and Parallel Systems (IJDPS)* (2011)

17. B.R. Nanjesh: *MPI Based Cluster Computing for Performance Evaluation of Parallel Applications, Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT).* JeJu (2013)

18. D.A. Mallón: *Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures, M. Ropo et al. (Eds.): EuroPVM/MPI 2009, LNCS 5759, Springer-Verlag Berlin Heidelberg* (2009)

19. M. Zhu: *Performance Evaluation of a Communication Optimization Model in Network-based Parallel Computing, Proceedings of International Workshops on Parallel Processing* (2000)