Massively Parallel Simulations of Agent-based Spatial Interaction: A Many-core Computing Approach with Spatial Big Data

Zhaoya Gong
School of Geography, Earth and Environmental Sciences
University of Birmingham
Birmingham, UK
Z.Gong@bham.ac.uk

Wenwu Tang
Department of Geography and Earth Sciences
University of North Carolina at Charlotte, USA
WenwuTang@uncc.edu

Jean-Claude Thill
Department of Geography and Earth Sciences
University of North Carolina at Charlotte, USA
Jean-Claude.Thill@uncc.edu

Abstract

Big spatial data is characterized by not only large data volumes but also high velocity, which poses challenges for data processing systems. This study focuses on addressing the issue of deficiency in parallel processing of high-velocity spatial big data that features fast changing patterns of spatial dependency over time. We propose to use a many-core architecture, Many Integrated Core (MIC), to the resolution of this issue. An agent-based spatial interaction model is customized to simulate massive distributed interactions and corresponding changing patterns over time steps and to mimic a course of real-time data processing with a certain velocity. Two groups of experiments are designed to evaluate the proposed approach. Our experimentation reveals that MIC enables superior model scalability and that MIC-supported models are only subjected to moderate influence on their performance with respect to changing model characteristics.
Keywords: MIC, many-core, spatial big data, high velocity, spatial interaction, ABM.

1 Introduction

Spatial interactions are omnipresent in a wide range of geographic phenomena, ranging from individual communication and information exchange to transportation flows and human migration at a more aggregated level, and from trading between regions and countries on the earth to the gravitational interaction between celestial bodies at coarser granularity. Agent-based models (ABMs) are well suited to capture the complexity of spatial interaction systems, because they characterize the spatiotemporal dynamics of global patterns emerging from local processes involving spatially distributed entities and the feedback of global-level emergence to the development of local interactions. Previous research has been dedicated to improving the scalability of spatial ABMs fed with big spatial data by taking advantage of various parallel computing resources in the emerging cyberinfrastructure (Gong et al., 2013; Gong et al., 2017). Specifically, the special parallel paradigm of Non-Uniform Memory Access (NUMA) has been proposed to tackle the uneven distribution of computing overhead caused by the heterogeneous patterns of spatial interactions. The known structure of interaction patterns and its constancy are assumed, so that the optimized decomposition and allocation of spatial domains across computing resources can be performed to minimize the data access cost and thus to enhance scalability. However, big spatial data feature not only large data volumes but also high velocity, such as real-time traffic information or social media feeds; this means that data streams come in so fast with constantly changing longitudinal patterns that are almost impossible to predict. This new challenge points out the deficiency of existing parallelization paradigms or approaches whose performance gain stems from determining the optimal parallelization strategy, given an interaction pattern; yet, it cannot keep up with the pace of pattern change of incoming data.

In this study, we propose to employ a new many-core co-processing platform, Many Integrated Core (MIC) coprocessors (Jeffers and Reinders, 2013), to resolve the performance issue for the parallelization of spatial interaction ABMs with high-velocity spatial big data. The MIC platform provides dedicated enhancements for non-homogenous data access without the interference of external optimization strategies for data access cost. This allows for a simplified external parallelization strategy that focuses on straightforward domain decomposition and load balancing, and may significantly reduce the extra effort in modifying and optimizing existing codes. By comparing the proposed paradigm applied to an ABM of spatial interactions with the previous NUMA paradigm, we demonstrate the superiority of the MIC platform over the other in terms of model scalability and performance.

2 Agent-based spatial interaction models

We use a hypothetical agent-based model to simulate the interaction between spatially situated individuals. Specifically, the interaction between individuals is reflected by an opinion exchange process as that in the bounded confidence model (Weisbuch et al., 2002). That is, individual agents (e.g., persons or households) are spatially distributed in a grid-based landscape. Each cell in the grid accommodates only one agent. Each agent will identify a neighbour to exchange opinion regarding an arbitrary topic via a neighbourhood setting. However, this communication is bounded by an assumption that, when two agents hold extremely different values, no opinion exchange occurs. Therefore, successful opinion exchange takes place based on a learning function (1), only when two agents hold similar enough opinions as set by a threshold σ:

$$W_i' = \begin{cases} (1-\rho)W_i + \rho W_i & if\ |W_i - W_j| \le \sigma \\ W_i & otherwise \end{cases} \quad (1)$$

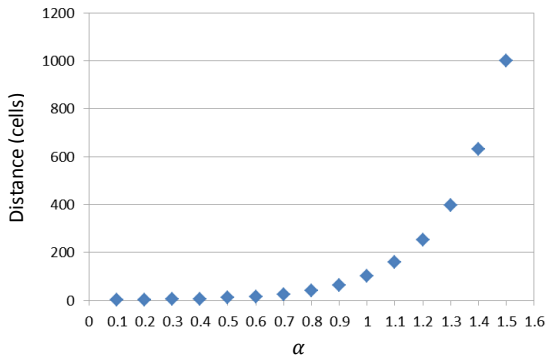
where W_i and W_j represent the current opinion values of agents i and j, W_i' is the new opinion value of agent i after communication, and ρ is the learning rate.

The temporal dimension of the opinion exchange follows an iterative process, where a series of time steps mimics a certain velocity of real-time data processing. At each time step, each agent has one opportunity to initiate an interaction with an identified neighbour. To create a different data pattern for each time step, agents follow a random sequence of interactions, which is realized via a shuffling mechanism at the beginning of every time step. Furthermore, we use a nondeterministic approach for neighbour identification, which increases the variation of data patterns over time steps. With the assumption of distance dependency, a circular neighbourhood is defined around each agent after the polar coordinate system. A distance-decay function (2) is used to relate the distance D_{ij} between an agent i and its neighbour j to the probability that i identifies j as its neighbour P_{ij}.

$$P_{ij} = D_{ij}^{1/-\alpha} \quad (2)$$

where α is a decay coefficient. Figure 1 demonstrates the effect of α on distance with $P_{ij} \le 0.01$. In other words, varying parameter α allows us to change the radius for neighbour search. During simulation, P_{ij} is randomly generated from a uniform distribution, and then D_{ij} can be determined by inverting equation (2) with a given parameter value for α. Together with a randomly selected azimuth between 1° and 360°, a neighbour can be identified using the polar coordinate pair (distance, azimuth). Figure 2 presents the model mechanism of our spatial interaction ABM via a flow chart (Grimm et al., 2006).

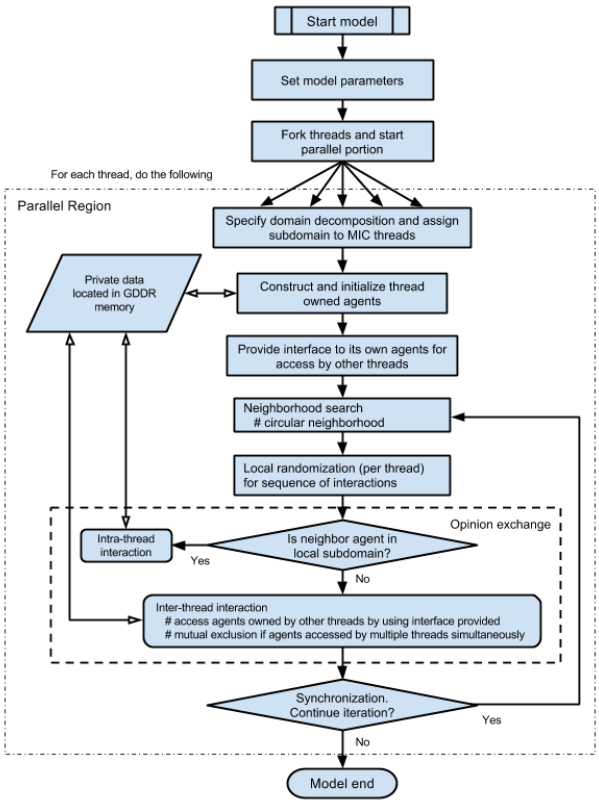
Figure 1: The relationship between distance and α with $P_{ij} \le 0.01$



3 MIC platform for parallel ABMs

Intel's MIC architecture is a many-core coprocessor platform that assists the main processors (i.e., CPUs) for certain types of computing tasks (e.g., floating point and graphics) in order to achieve better efficiency (Jeffers and Reinders, 2013). Another popular coprocessor platform is Graphical Processing Units (GPUs), which is dedicated to graphics processing in nature. As a many-core platform, a MIC chip integrates up to 61 lightweight cores having a capability to provide 4 hardware threads on each core, which enables massive parallelism and high throughput. Compared to common multi-core processors, MIC supports smaller cores, more hardware threads, wider vector units and higher bandwidth for memory access. Due to MIC's architectural compatibility, it is much easier to port existing sequential and parallel codes running on multi-core processors to MIC comparing to the GPU platform that

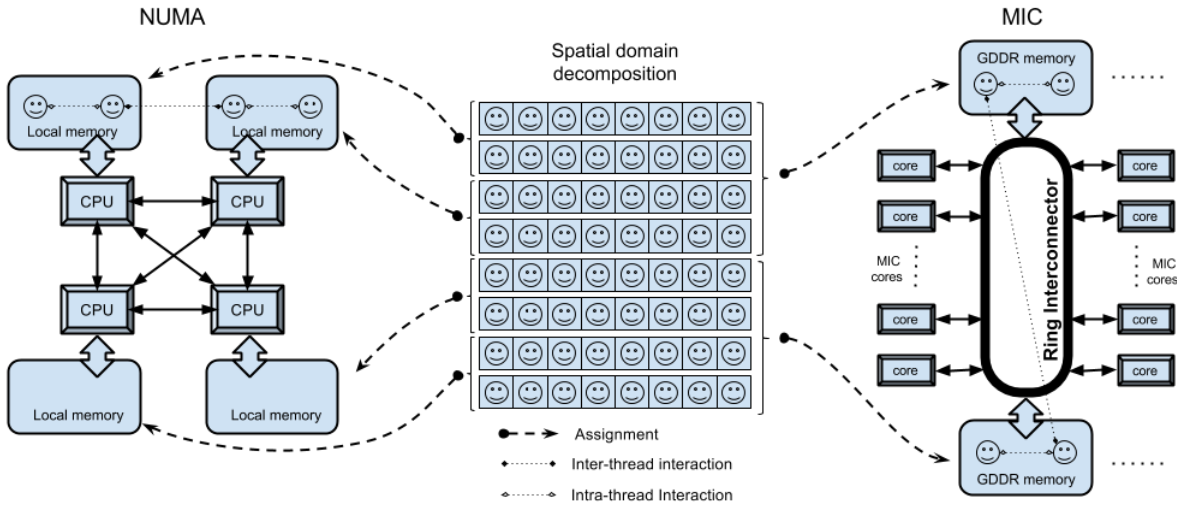
Figure 2: Flow chart of our parallel spatial interaction ABM



typically involves rewriting or heavily modifying existing code to fit new programming models.

Gong et al (2017) proposes multi-core platforms with NUMA architecture for parallel spatial ABMs with heterogeneous interaction patterns. The key to achieving a highly scalable performance for NUMA is that distributed and localized memories are attached to different processors, while local memories are also visible to remote processors (the left part in Figure 3). As a shared-memory system as well, MIC has distributed cores and GDDR memories (high-performance RAM with a high bandwidth designed for use in graphics cards) connected by a ring-shape, two-way circling, high-speed interconnection bus for data communication (the right part in Figure 3). A comparison of MIC and NUMA is shown in Figure 3. It illustrates that, from a model parallelization perspective, the landscape of our ABM is decomposed into spatial partitions and the partitions are then assigned to

Figure 3: Comparison of MIC and NUMA architectures for the parallelization of spatial interaction ABMs



different memory spaces for parallel computing (the middle part in Figure 3). On a NUMA platform threads generated by cores compute the partitions in their local memories, while on a MIC platform each thread computes one of the partitions which are automatically distributed and managed by the platform. For both platforms, interactions between agents located in the same partition are intra-thread interactions, while interactions between agents located in different partitions are inter-thread interactions. Intra-thread interactions are less expensive because data access occurs within one local memory bank. In contrast, inter-thread interactions are much more costly due to the overhead caused by data access between different memory banks. Detailed parallel processing of our spatial interaction ABM is presented in Figure 2.

4 Model implementation

Our models are implemented on an Intel Xeon Phi SE10P Knights Corner MIC coprocessor on a PCIe card, which is hosted by a compute node in the Stampede supercomputer system from the Texas Advanced Computing Center. This MIC coprocessor contains 61 cores (1.1 GHz) and 8 GB of GDDR5 memory. For comparison, the models are also implemented on a compute node from a smaller Linux cluster. This node has 32 CPU cores (AMD Opteron 2.0 GHz) and 64 GB of memory. Both implementations aim to exploit thread-level parallelism and are coded in C++ and OpenMP (Chapman et al., 2008). The latter is a standard specification for multithreading programs running on shared-memory platforms.

To assess the performance and scalability of our parallel models, two metrics, Speedup and Efficiency, are defined as follows (Wilkinson and Allen, 2004):

$$S = \frac{T'}{T},\ E = \frac{S}{R} \tag{3}$$

where T is the execution time for a baseline model (e.g., sequential model), T' is the execution time for a target parallel model, and R is a ratio between the number of computing units used by the target model to the number of computing units used by the baseline model. For example, if a baseline parallel model uses 2 threads and the target parallel model uses 32 threads, $R = 16$.

5 Experiments

To evaluate the performance of a MIC approach for the parallelization of our spatial interaction ABMs, experiments are designed to examine 1) the scalability of the parallel models on a MIC platform with a comparison to that for a NUMA platform, and 2) the impacts on the model performance when two spatial characteristics of the ABMs are changed. Throughout all experiments, at most 60 cores of the MIC coprocessor will be used with 4 threads running on each core. For the NUMA platform, only one thread per core will be used for its best performance. A straightforward horizontal decomposition strategy with balanced workload for each thread (Figure 3) is applied to parallel models for both platforms. This simplification is intentional in order to better single out the effect of the underlying platform on model performance.

5.1 Scalability comparison

We first examine the scalability of our parallel AMBs on both MIC and NUMA platforms as computing units utilized increase. For the convenience of comparison, we arrange a series of subsets of cores for both platforms to have the same number of treatments. The specifications regarding the number of cores/threads for all treatments are detailed in Table 1. This arrangement specifies that the baseline model for MIC uses 10 cores (40 threads) while for NUMA the baseline uses 1 core (1 thread). Therefore, the ratio R

(Equation 3) for each treatment can be calculated accordingly. Table 1 also shows the computing times for models supported by MIC and NUMA platforms. The model time is the portion of total program execution time, excluding the shuffle time (the aggregated time taken to do shuffling at the beginning of each time step during a model run). Thus, the model time allows us to focus on the aggregated time taken for spatial interactions.

Table 1: Comparison of MIC and NUMA in terms of model computing time (second) for different treatments (landscape size = 4000 x 4000 and $\alpha = 0.1$)

MIC			NUMA		
Threads/ Cores	*R*	Model	Threads /Cores	*R*	Model
40/10	**1**	**975.03**	**1/1**	**1**	**3591.51**
80/20	2	501.50	2/2	2	2469.88
120/30	3	323.30	4/4	4	1300.06
160/40	4	243.57	8/8	8	697.26
200/50	5	194.83	16/16	16	354.21
240/60	6	162.56	32/32	32	163.17

Note: the bolded row indicates baselines

Note that all model runs in Table 1 have used the same set of parameters for spatial characteristics, landscape size = 4000 x 4000 and neighbourhood radius $\alpha = 0.1$. To reduce the random effect, for each treatment 10 model runs are conducted and averaged results are obtained throughout all experiments in section 5.

Figure 4: Comparison of model efficiency for MIC and NUMA

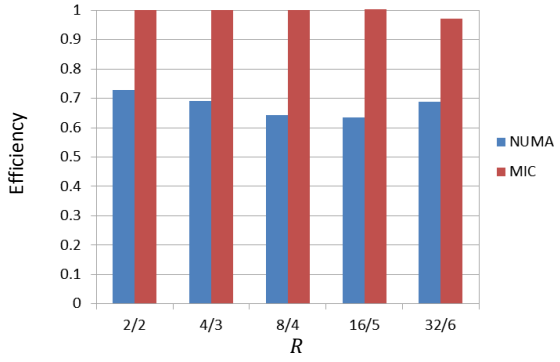


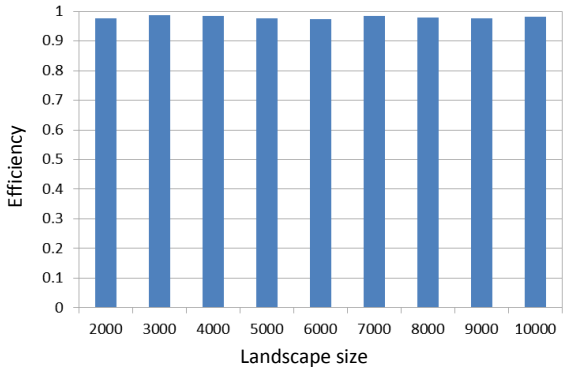
Table 1 shows that MIC and NUMA achieve similar performance (162.56 vs. 163.17) for fully loaded systems, though MIC is slightly faster. Based on Table 1, Figure 4 can be created to compare the scalability of MIC and NUMA in terms of the Efficiency metric. For NUMA, as the computing units increase from 2 to 16, model efficiency drops to 0.63 with an initial value of 0.73. Until the system is fully loaded with all cores (32 threads), the efficiency increases back to 0.68. In contrast, model efficiency metrics for MIC are equal or close to 1 (perfect efficiency) over all treatments. Note that when 60 cores (240 threads) are used, the efficiency (0.97) is slightly lower than 1, which can be explained by the overheads generated from a large volume of inter-thread interactions between many partitions (240). In sum, the overall performance of model for NUMA is in the range of 0.63 to 0.73, which is much lower than that for MIC across all treatments. This comparison confirms a superior scalability of the parallel models supported by MIC than that for NUMA.

5.2 Impacts of spatial characteristics

We turn to examine here how the MIC platform responds to the impacts of spatial characteristics of spatial interaction with respect to model performance. Specifically, two characteristics are under consideration, namely, landscape size and parameter α for neighbourhood radius.

Given a grid landscape, a series of sizes are included in the experiments, ranging from 1,000 x 1,000 to 10,000 x 10,000 with a 1,000 x 1,000 interval. $\alpha = 0.9$ is set for all treatments. The aim is to examine how efficiently the model performs as the volume of interactions increases with an enlarging landscape size. We take a similar approach as the calculation of Efficiency for scalability in (3). The model with a size of 1,000 x 1,000 is used as a baseline and the ratio R is calculated as the ratio of the size for a target model to the size of the baseline. Thus, we can obtain a metric indicating the efficiency of model performance with respect to landscape size, which is reported for each treatment in Figure 5. It illustrates that the parallel models exhibit very high performance with efficiency metrics close to 1 over all treatments and this performance is quite stable as landscape size increases.

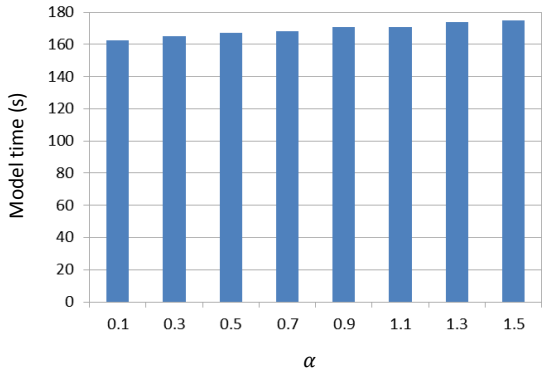
Figure 5: Model efficiency with respect to landscape size for MIC



The second spatial characteristic, α, determines the radius to search a neighbour in spatial interactions. As reflected by Figure 1, larger α means higher probability to identify a neighbour in a different partition, which may result in inter-thread interaction and larger overheads for data access. The experiments use a series of α values from 0.1 to 1.5 with an interval of 0.2. Landscape size of 4,000 x 4,000 is fixed for all

treatments. The model time (defined previously) for each treatment is reported in Figure 6. With larger α values, model time increases from 163 to 175 seconds, which confirms our expectation that larger neighbourhood radii lead to more inter-thread interactions crossing partitions and thus creating more overheads for remote data access. However, increase in the model time is relatively low compared to the entire model time. Therefore, with the support of MIC, the impact of neighbourhood radius on model performance is moderate.

Figure 6: Model execution time (second) with varying α for MIC



6 Conclusions and future work

This study aimed to tackle the challenge of deficiency in parallel processing of high-velocity spatial big data that features fast changing patterns of spatial dependency over time. We proposed to employ a many-core architecture, MIC, to the resolution of this issue. An agent-based spatial interaction model was customized to simulate massive individual interactions and the changing interaction patterns over time steps and to mimic a course of real-time data processing with a certain velocity. We designed two groups of experiments to evaluate the proposed approach in terms of model scalability and performance. Compared to the NUMA architecture, MIC achieves superior scalability in handling the temporal variation of spatial interaction patterns even with a simple parallelization strategy. In addition, the model's spatial characteristics pertaining to the volume and range of interactions across space exerts moderate impact on model performance, which is at an acceptable level. The assessments validate the high efficiency of the MIC architecture in the real-time processing of spatial big data. This can be attributed to its dedicated enhancement for non-homogenous data access, which could significantly reduce extra efforts in parallelizing and optimizing existing codes. Future work may include further evaluation on how the spatial characteristics of interactions affect the scalability of model efficiency. Comparison of MIC to other many-core architectures, such as GPUs, is also worth more exploration with respect to the real-time processing of spatial big data.

Acknowledgement

Support from US NSF XSEDE Supercomputing Resource Allocation (SES170007) "Accelerating and enhancing multi-scale spatiotemporally explicit analysis and modelling of geospatial systems" is acknowledged.

References

Chapman, B., Jost, G. and Van Der Pas, R. (2008) *Using OpenMP: portable shared memory parallel programming*. Cambridge, MA: MIT Press.

Jeffers, J. and Reinders, J. (2013) *Intel Xeon Phi coprocessor high-performance programming*. Newnes.

Gong, Z., Tang, W., Bennett, D.A. and Thill, J.C. (2013) Parallel agent-based simulation of individual-level spatial interactions within a multicore computing environment. *International Journal of Geographical Information Science*, *27*(6), pp. 1152-1170.

Gong, Z., Tang, W. and Thill, J.C. (2017) A graph-based locality-aware approach to scalable parallel agent-based models of spatial interaction. In: Griffith, D. A., Chun, Y. & Dean, D. J. (eds.) *Advances in Geocomputation.* Springer, Cham, Switzerland, pp. 405-423.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S.K., Huse, G., Huth, A., Jepsen, J.U., Jørgensen, C., Mooij, W.M., Müller, B., Pe'er, G., Piou, C., Railsback, S.F., Robbins, A.M., Robbins, M.M., Rossmanith, E., Rüger, N., Strand, E., Souissi, S., Stillman, R.A., Vabø, R., Visser, U., and DeAngelis, D.L. (2006) A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1-2), pp. 115-126.

Weisbuch, G., Deffuant, G., Amblard, F. and Nadal, J.P. (2002) Meet, discuss, and segregate!. *Complexity*, *7*(3), pp. 55-63.

Wilkinson, B. and Allen, M. (1999) *Parallel Programming*. Upper Saddle River, NJ: Prentice hall.