Parallel execution of space-aware applications in a Cloud environment

Franco Cicirelli, Agostino Forestiero, Andrea Giordano, Carlo Mastroianni, Giandomenico Spezzano
ICAR-CNR, Rende (CS), Italy
Email: {cicirelli,forestiero,giordano,mastroianni,spezzano}@icar.cnr.it

***Abstract*—This paper analyzes and evaluates the strategies and implications related to the execution of parallel algorithms on a distributed Cloud infrastructure, with the focus on an important class of applications for which the execution is performed on spatial data, dislocated on a bidimensional territory. Applications of interest cover a wide spectrum ranging from Internet of Things to social sciences, geology, swarm-inspired computation etc. The territory is partitioned into regions, and regions are assigned to parallel computational nodes to speed up the execution. Parallel nodes are aligned through the exchange of messages in order to ensure a coherent and efficient execution. The paper offers an analysis of the parallelization cost in this context, especially in terms of communication overhead, which is essential to estimate the impact of porting the computation onto a Cloud environment. More in particular, the paper evaluates two different strategies for space partitioning, i.e., linear partitioning and bidimensional partitioning, with a specific focus on scalability analysis, and compares the two strategies when both options are exploitable.**

***Keywords*—space-aware applications; communication overhead; Cloud; Internet of Things; parallel computation; multi-agent algorithms**

I. Introduction

In the last few years, increasing attention is devoted to the field of the so called “Internet of Things” (IoT), an emerging paradigm built upon the research and development advances in a wide range of key areas including wireless and sensor networks, mobile and distributed computing, embedded systems, agent technologies, autonomic communication, Cloud computing. The variety of involved application domains is wide [1]: transportation and logistics, smart electrical grids, big data and business analytics, intelligent management of “smart cities”, social sciences, etc.

The basic idea of the IoT is a pervasive presence around us of a variety of things or objects such as RFID tags, sensors, actuators, mobile phones, etc. which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals [2]. One of the most important issues of the IoT is the huge amount of data generated from the devices: Gartner study forecasts that the IoT will reach about 26 billion of objects by 2020 [3].

This paper is partially financed by the MIUR project PON03PE_00050_2, “Sistemi Domotici per il servizio di brokeraggio energetico cooperativo”

Due to the very nature of these domains, data and objects are often strictly related to the space or territory on which they are defined and used: for example, environmental information extracted from sensors, data inherent to the neighborhoods and residential units in a city, data generated by Internet geographical domains, etc. It is then natural to manage such data through the use of computing entities distributed in the territory, in order to perform computation as close as possible to data sources and increase the performance.

Many space-aware applications require massive data storage, huge processing speed and large broadband networks to enable real-time decision making. Cloud computing provides an ideal back-end solution for handling the data deluge and the unprecedented number of entities. In general it is not feasible or convenient to bring computation to a single Cloud infrastructure, e.g., a big centralized data center. A better support to tackle mobility and geo-distribution of data, embrace location awareness and ensure low latency, can be provided by a variant of Cloud, referred to as Fog Computing [4] [5], which is composed by a number of distributed Cloud facilities located close to data sources, i.e., a cloud close to the ground.

A large number of parallel applications, and almost all those related to space-aware domains and the IoT paradigm, cannot be addressed by “embarrassingly parallel” computation [6], for which the parallel tasks do not need to exchange data during computation. For example, a smart city application typically requires that the events occurred in a city neighborhood are frequently communicated to the adjacent neighborhoods.

The objective of this paper is to quantify the communication burden of space-aware parallel applications. We analyze the very frequent case in which the territory is partitioned into regions. The regions, along with the corresponding entities, data and computation, are assigned to distributed nodes of a Cloud/Fog infrastructure. Two alternative solutions for space partitioning are considered in this work, i.e., linear partitioning and bidimensional partitioning, and the communication burden is analyzed when varying some basic parameters, e.g., the number of regions and the type of horizontal and vertical cuts of the territory. Beyond analyzing the two scenarios separately, we also compare the performances of linear and bidimensional partitioning, in order to help individuate the most efficient option when both solutions are admissible.

This paper tackles the mentioned issues for a general and widely adoptable context, and the outcome will also be used in the specific scenario addressed by the Italian project "Cooperative Energy Brokerage Services", financed by MIUR, whose main objective is to build a Cloud infrastructure, integrated with a sensor network, that will help improve the efficiency of electricity and gas distribution in a urban context.

The rest of the paper is organized as follows: Section II discusses some relevant state-of-the-art on the efficient execution of parallel applications in a Cloud environment, with a specific focus on communication issues. Section III introduces the issues related to territory partitioning, and discusses the methodologies used to ensure the correctness and efficiency of parallel computation in this context. Section IV illustrates how the communication burden is related to the size of border areas that overlap adjacent regions, analyzes and compares the burdens related to the cases of linear and bidimensional partitioning. Finally, Section V concludes the paper.

II. Related Work

The transmission of data, based both on the network fabric and its efficient support in the virtualization layer, heavily influences the scalability of parallel applications on Cloud infrastructures [7], and the estimation of the amount of data that needs to be transmitted among the computing nodes is of paramount importance [8].

Although the issue of inter-node communication in parallel Cloud applications is topical, especially for the types of application discussed in this paper, the field has not been the object of many research works so far. The execution of parallel/distributed applications on Cloud/Fog platforms has been analyzed by many recent papers, most of which are, however, focused on the case of embarrassingly parallel applications [6]. Such applications can be tackled by master/slave paradigms such as MapReduce, and implemented through Cloud technologies such as Hadoop, Dryad, DryadLINQ and CGL-MapReduce. In [9], it is recognized that communication is an important issue for parallel Cloud applications, and different communication patterns are evaluated. However, the authors notice that none of the major Cloud providers offer information about the specific network interconnections used among the machines, and it is impossible to determine the network latency or distance between nodes. This can provide challenges to communication-intensive applications, and it is a major disadvantage compared to on-premises clusters. Moreover, the lack of precise information on this aspect may hinder the accurate estimation of the impact and costs of porting a parallel application to the Cloud.

In Cloud environments that span multiple data centers or sites, applications may experience performance degradation due to high latency and low bandwidth data transmission induced by the use of communication links of wide area networks. This degradation is mainly caused by non-optimal implementations of the mechanisms adopted for message exchange. In [10] a new approach is proposed to increase the performance and scalability of distributed Cloud applications, based on an optimized algorithm that refines the Open MPI standard. Other research efforts [11] [12] aim to minimize the total traffic in a data center by allocating virtual machines (VMs) that experience high inter-traffic on the same host or cluster. The goal is to improve the initial placement of the VMs that host the tasks belonging to a parallel application. The approach illustrated in [13] focuses on the case that the demand for bandwidth and the communication patterns of parallel applications are highly dynamic, and proposes a communication-aware and energy-efficient scheduling to reactively reschedule the placement of VMs in real time. In particular, the approach aims to reallocate the VMs that generate the largest amount of network traffic. The issue is also tackled by [14] through the adoption of the public/subscribe paradigm: Azure's storage services are used to transfer data, while a broker relays notifications to interested parties.

The state of the art in this field, of which this section has offered an excerpt, clearly shows a growing interest on addressing the issues related to the execution of non embarrassingly parallel applications on a Cloud environment. Unfortunately, the analysis and design of the aspects associated with the exchange of data among parallel nodes is in general a difficult task, and cannot be tackled with universally applicable solutions. However, specific approaches can be tailored to harness the peculiar characteristics of different application domains. This paper focuses on an important class of applications, i.e., space-aware applications. A general methodology is presented for parallelizing the execution by associating computational nodes with the portions of the involved territory, and the communication burden related to different parallelization strategies is analyzed in detail.

III. Parallelizing Execution through Space Partitioning

Many problems related to several application domains, ranging from the Internet of Things to geology, biology and social sciences, require the explicit definition and management of bidimensional data structures representing and reproducing physical spaces or territories. In this paper, the territory is managed as a bidimensional grid of *cells*. Each cell has a position and can contain *active* and *passive* entities. An active entity, also referred to as *agent* in the following, is able to perform autonomous computation, explore and modify the territory and move around it. Passive entities represent the data upon which active entities perform the computation. The abstraction adopted here, consisting of a territory populated by active and passive entities, is very general and allows to support different kinds of computing paradigm like mobile agents [15]–[17], cellular-based automata [18] and bio-inspired computation [19]–[21].

The *operational radius* – referred to as r in the following – delimits the boundary of agents operations, i.e., the

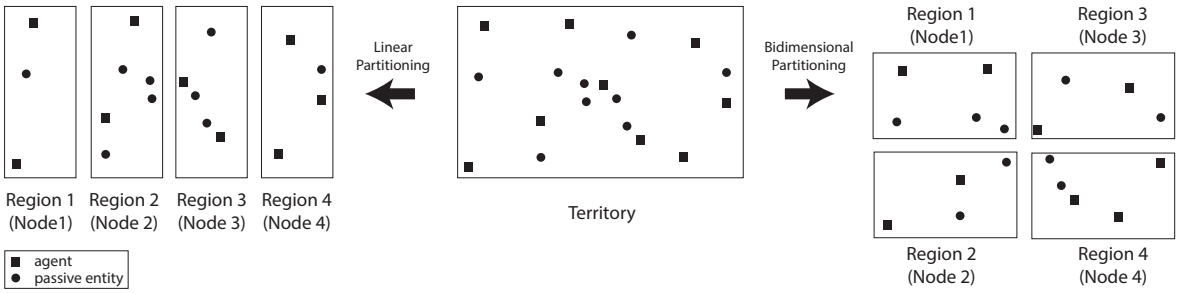

Figure 1. The territory partitioned into regions which are associated with parallel computing nodes. Two alternative types of partitioning are shown, linear and bidimensional.

operational area. Hence, an agent can perceive and perform modifications only within the portion of the territory included in its operation area, centered in the position of the agent.

A natural way of parallelizing the execution of algorithms working on spatial data is to partition the territory into *regions* and assign each region, along with the contained entities, to a *computing node* that will be in charge of performing the associated computation. Partitioning favors system scalability in that as the size of the territory increases, more computing nodes can be used to speed up the execution. A territory can be partitioned through either a *linear* or a *bidimensional* schema, as shown in Figure 1.

In some application scenarios, the computation related to each region evolves independently from other regions. In such cases, the application is defined as "embarrassingly parallel" [6]. In the scenario considered here, a space-aware application results to be embarrassingly parallel when the operational area of each agent is completely contained in a single region and agents do not migrate among regions. These conditions ensure that nodes do not exchange data during computation, and that the computation carried out in a node does not affect the computation at any other node. In most cases, however, applications are not embarrassingly parallel. Therefore, inter-node communication is required and a proper alignment policy is needed, i.e., a policy that aligns the computation on the nodes and ensures a coherent execution of the application in the parallel context. For the sake of simplicity, we consider the frequent case in which the computation is *step-based*, i.e., the nodes align among each other at each step. However, the outcomes of our study can also be applied when different alignment policies are adopted.

Figure 2 shows a scenario where the operational area of an active entity falls into two different adjacent regions (if linear space partitioning is adopted) or into four different adjacent regions (in the case of bidimensional space partitioning). In such cases, the application is not embarrassingly parallel, because an agent may require to access and/or manipulate *remote* data, i.e., data located in different computational nodes.

Access to remote data must be *correct* and *efficient*. Correctness, in this context, means that agents always work with updated information. For example, a correctness violation

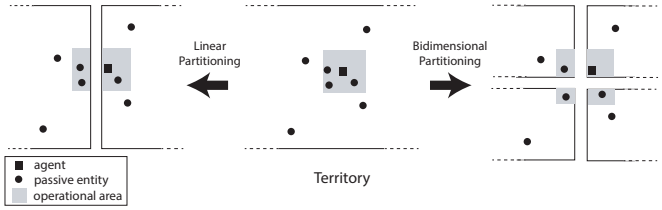

Figure 2. Partitioning of a territory and of an operational area

occurs when a piece of data is modified at steps t_1 and t_2, with $t_2 > t_1$, and an agent, at a successive step, accesses the piece of data in its t_1 version. To avoid such an incorrect access to data, a step-by-step duplication mechanism can be adopted, which consists in replicating the edge areas of adjacent regions. Such areas, referred to as *borders*, are kept aligned by exchanging at each step *update messages* between the adjacent computing nodes, i.e., the nodes that manage the corresponding adjacent regions. This data exchange ensures that data is always updated at the last step. Replication of borders data is a valuable solution also from the efficiency point of view, as it ensures that active entities access data only through local operations. Indeed, agents work with local replicas of data and do not need to engage network operations to access remote data. This increases the efficiency of operations on data and helps to reduce inter-node communications, thus improving performances.

The border area of a region is composed of two distinct parts: the *local border* and the *mirror border*. Figure 3 and Figure 4 show the borders in the cases of linear and bidimensional space partitioning. The local border is managed by the local node and its content, i.e., the active and passive entities, is replicated in the mirror border of the adjacent nodes. At each step, all the modifications occurred in a local border are gathered and transmitted to the adjacent nodes. For example, information about the updates occurred in the local border of Node 1 of Figure 3 are sent to Node 2, which applies the updates in its mirror border. Analogously, information in the mirror border of Node 1 is aligned with the updates occurred in the local border of Node 2. In the case of bidimensional partitioning, see Figure 4, an update message may be sent to more than one adjacent nodes. Figures 3 and 4 also show that

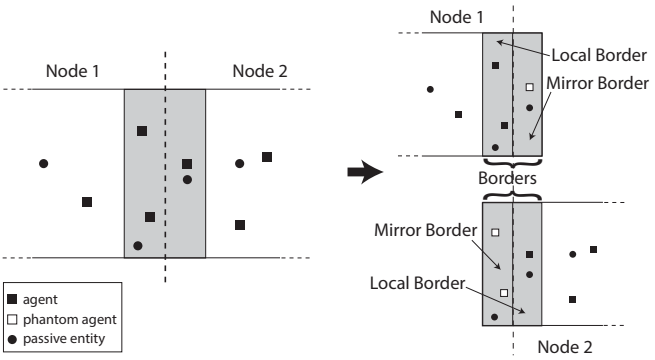

Figure 3. Border areas of two adjacent nodes in the case of linear partitioning

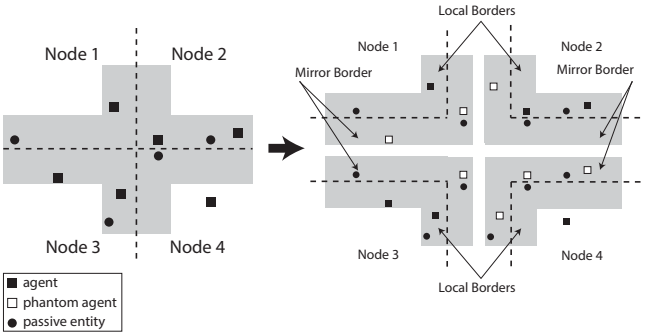

Figure 4. Border areas of four adjacent nodes in the case of bidimensional partitioning

the agents located in a border area are mirrored by means of *phantom* agents (i.e., agents that hold the same information of original agents but are not allowed to perform computation), while passive entities are simply duplicated.

It should be noticed that territory partitioning may also originate concurrency issues, when active entities allocated on adjacent regions operate concurrently on the same passive entities. A methodology to cope with such issues has been presented in [17] and [22].

IV. Analysis of the communication burden

In this section we analyze the amount of data that is involved in the execution of non embarrassingly parallel applications on a Cloud platform. As explained in the previous section, we cope with the frequent case in which the parallelization is achieved by partitioning a territory into regions, assigned to different computational nodes, which need to exchange data during execution. We first analyze the case of linear partitioning, then the case of bidimensional partitioning, and then we provide a comparison of the two approaches when both options are available.

Data must be exchanged to keep regions informed about the updates occurred in the adjacent regions, more specifically in the region borders, as illustrated in Section III. Therefore, the area of the borders are a good proxy variable[a] for the estimation of the communication burden. Indeed, if data and updates are uniformly distributed over the territory, the border area is proportional to the amount of data that is updated and needs to be exchanged. In cases where the distribution is not uniform, the communication burden can be derived by considering both the border areas and the type of data and updates distribution.

The scenario under consideration consists of a toroidal rectangular territory with horizontal size equal to L space units and vertical size equal to H space units[b]. The territory is partitioned into N regions through a number of horizontal cuts, C_l, and a number of vertical cuts, C_h. Each region has horizontal size $l = L/C_h$ and vertical size $h = H/C_l$, and is assigned to an associated computational node for execution.

While in this paper we focus on the communication burden, of course the time needed for parallel execution is also of outmost importance. The execution time of a parallel application computed on N nodes, T_N, can be expressed as [23]:

$$T_N = \frac{T_1}{N} + T_O = \frac{T_1}{N} + T_{comm} + T_{idle} + T_{conf} \qquad (1)$$

where T_1 is the sequential time, i.e., the time to execute the computation on a single node, and T_O is the overhead time added by parallelization. The overhead time is the sum of three main contributions related, respectively, to the time needed for building and transmitting data between nodes (T_{comm}), the idle time experienced by faster nodes that need to wait for slower nodes (T_{idle}) and the time needed to solve conflicts, i.e., manage contentions on shared data (T_{conf}).

In the type of scenario considered in this paper, the component T_{comm} is often the most important because it is necessary to exchange, among computational nodes, a significant amount of data related to the different regions in which the territory is partitioned. Of course, T_{comm} is strictly related to the communication burden discussed and analyzed in this section. In particular, T_{comm} is related to the time needed by a single node to transmit update messages to its adjacent nodes. Since update messages are concurrently exchanged at each step of the computation, T_{comm} is given by the maximum value of the communication times experienced by single nodes. In general, a reduction of the amount of exchanged data implies a reduction of T_{comm}. The detailed analysis of the overhead time, which must include also the evaluation of the other two components, T_{idle} and T_{conf}, is out of scope of this paper. This issue is currently under analysis.

[a]For the sake of clearness, here we consider the meaning of the term "proxy" as it is used in statistics, i.e., as a variable that helps to estimate another variable which is more complex to compute.

[b]Depending on the specific application, a space unit can be a meter, a kilometer etc.

A. Linear space partitioning

To analyze the case of linear space partitioning we assume, without loss of generality, that the space is partitioned through vertical cuts. Let us consider the sum of the areas of the left and right borders of a single region, shown in dark grey in Figure 5. This quantity, denoted as B_l and measured in squared space units (s_u^2), is taken as a proxy variable for the communication burden in charge of a single node. For the sake of simplicity, we refer to this quantity simply as "communication burden", but it is implicitly taken into account that, to compute the actual amount of exchanged communication, the amount of data transmitted per space unit must be considered. For the case of linear space partitioning, the communication burden of a single node is equal to $B_l = 2hr = 2Hr$. The overall communication burden, denoted as B_l^{tot}, is defined as the sum of the burdens of the N nodes, and is equal to:

$$B_l^{tot} = 2HrN \tag{2}$$

It is noticed that B_l and B_l^{tot} depend on the vertical size H, while they do not depend on the horizontal size L. Therefore, to minimize the value of the communication burden, it is convenient to consider the shorter side of the territory as the vertical side, so that $H <= L$. Moreover, the communication burden of a single node, B_l, does not depend on the number of involved computational nodes. From the point of view of the analysis of the execution time, this means that using more nodes allows to decrease the parallel time, i.e., the first component of the execution time in expression (1), T_1/N, whereas the communication overhead T_{comm} is not affected.

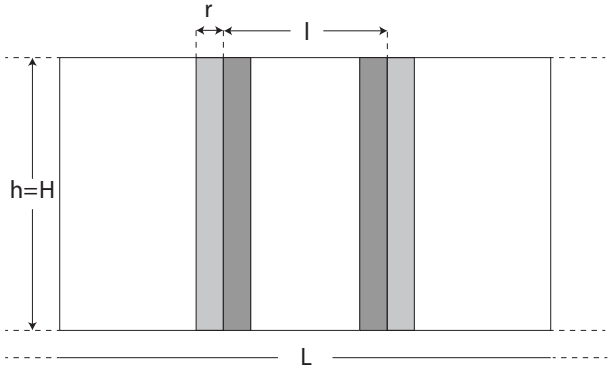

Figure 5. Linear space partitioning.

As an illustrative example, Figure 6 shows the value of the overall communication burden B_l^{tot} vs. the number of nodes in a scenario where the value of H is equal to 200 space units.

B. Bidimensional space partitioning

In the case of bidimensional space partitioning, not all the borders are replicated on a single adjacent region, as in the linear case: some borders must be replicated on two or more regions, depending on how the space is partitioned.

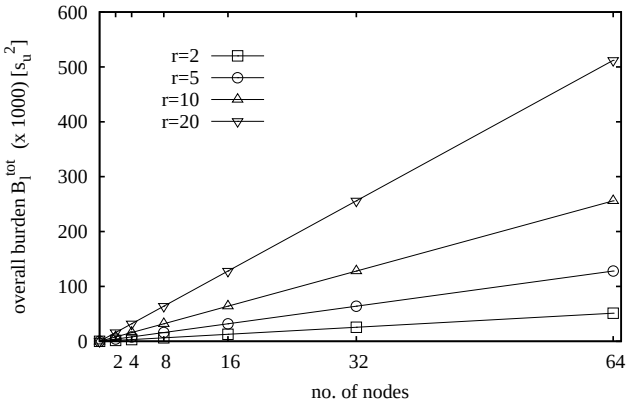

Figure 6. Linear space partitioning: overall burden vs. the number of nodes/regions, for different values of the operational radius.

For example, Figure 7 pictures a grid-like space partitioning. The borders of the central region are distinguished and labeled with numbers "1" and "3", depending if the corresponding area is replicated on one or three adjacent regions, respectively. In this scenario, the communication burden should be computed by properly weighing the contributions of the two kinds of border area. Specifically, the sum of the four areas labeled with "3", denoted as A_3, is equal to A_3=$4r^2$. The sum of the areas labeled with "1", denoted as A_1, is equal to A_1=$2r(l+h)-8r^2$. When computing the communication burden, the quantity A_3 is multiplied by three, since the corresponding areas must be replicated on three regions. The communication burden of a single region is then given by:

$$B_b = A_1 + 3A_3 = 2(l+h)r + 4r^2$$

and the overall communication burden for the N regions is:

$$B_b^{tot} = 2N(l+h)r + 4Nr^2 \tag{3}$$

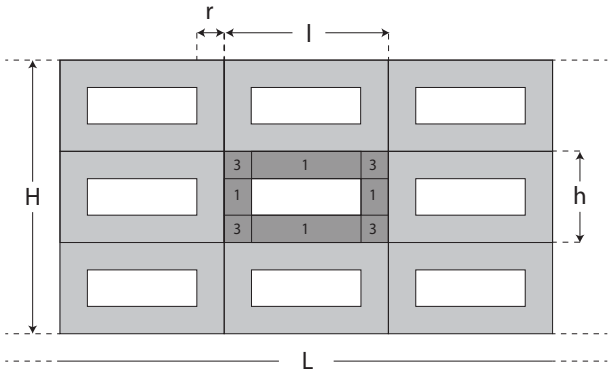

Figure 7. Grid-like bidimensional space partitioning.

Another option for bidimensional space partitioning is pictured in Figure 8. In this case, thanks to the different geometrical alignment of regions, border areas must be replicated

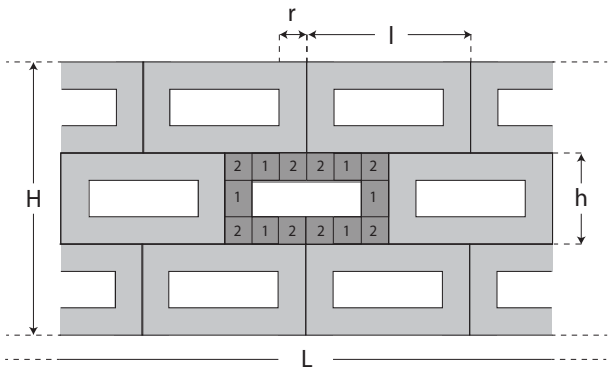

Figure 8. Bidimensional space partitioning with an alternative alignment of regions.

on only one or two adjacent regions; such border areas are labeled with "1" and "2" in the figure. For each region, the sum of the areas that must replicated on two adjacent regions is equal to $A_2 = 8r^2$, while the sum of the areas replicated on a single adjacent region is equal to $A_1 = 2r(l+h) - 12r^2$. The communication burden of a single region is defined by properly weighing the two components:

$$B_b = A_1 + 2A_2 = 2(l+h)r + 4r^2$$

and the overall communication burden is:

$$B_b^{tot} = 2N(l+h)r + 4Nr^2$$

A bit surprisingly, it results that the communication burden does not depend on which bidimensional space partitioning is adopted, i.e., the one depicted in Figure 7 or the one shown in Figure 8. Hence, the following analysis, focused on communication and data transmission aspects, does not depend on the chosen option. Nevertheless, it should be remarked here that the choice of space partitioning does have an impact on the execution time, and specifically on the time needed to synchronize the regions among each other, referred to as T_{idle} in expression (1). Indeed, in synchronization phases the computation on a region must wait until the updates from the adjacent regions (more specifically from the corresponding borders) have been received. In the scenario of Figure 7, a region must synchronize with eight adjacent regions, while in the scenario of Figure 8 synchronization is limited to six regions. From this point of view, the second option should then be preferred.

In the case of bidimensional space partitioning, as opposed to the linear partitioning scenario, it is worth analyzing both the single node burden and the overall burden, as the two quantities impact differently on different metrics, as explained in the following. Figure 9 shows the communication burden of a single node in a scenario where a 200x200 territory is partitioned into a number of nodes ranging from 4 to 64, assuming C_h=C_l. Different curves correspond to different values of the operational radius r. The figure shows that the single node communication burden decreases when the number of regions increases. Therefore, using more nodes allows to decrease not only the parallel time, i.e., the first component of the execution time in expression (1), T_1/N, but also the second component, i.e., the communication overhead T_{comm}. The reason for the latter outcome is that the amount of data transmitted between two adjacent regions decreases, since the involved border areas are smaller, and the time needed to exchange such data decreases as well.

On the other hand, Figure 10 shows that the overall communication burden increases with the number of nodes, because the sum of the border areas increases. As a consequence, it is possible to tune the appropriate number of nodes depending on the performance goals and/or the costs imposed by the Cloud provider. A larger number of nodes may be convenient if the main objective is to reduce the execution time or when the main cost is related to the time interval for which Cloud resources are rented. Conversely, it may be convenient to limit the number of nodes when the amount of transmitted data is the main cost or it is bounded by bandwidth constraints.

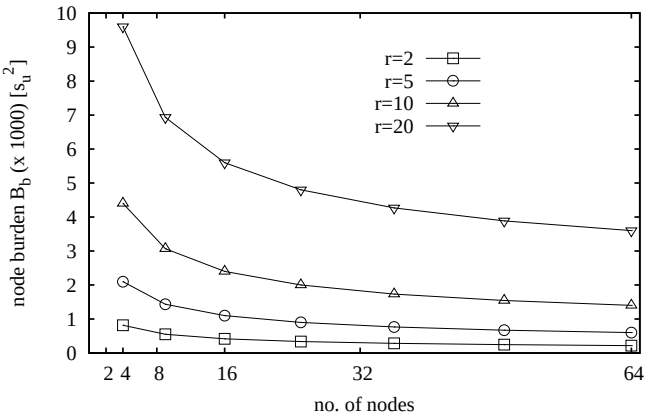

Figure 9. Bidimensional space partitioning: node burden vs. the number of nodes/regions, for different values of the operational radius.

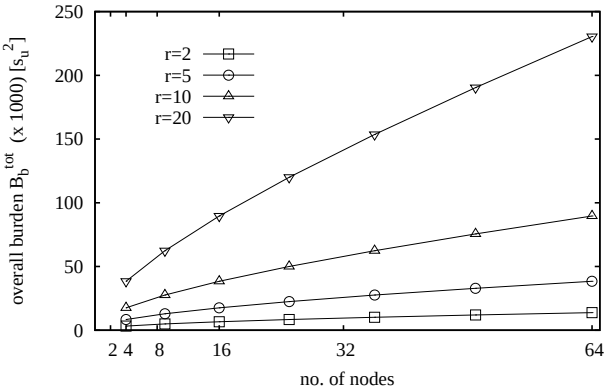

Figure 10. Bidimensional space partitioning: overall burden vs. the number of nodes/regions, for different values of the operational radius.

In the following, we focus on the impact that the shape of the regions (i.e., how *stretched* the rectangles are) has on the communication burden. In fact, for a given territory, while the area of a region only depends on the number of regions

N, the communication burden B_b^{tot} explicitly depends on the semiperimeter $(l+h)$, see expression (3), then on the regions shape. In the following it is shown that the minimum overhead is obtained when regions are square-shaped.

We express B_b^{tot} in terms of l, by using the substitution $h = \frac{S}{l}$, where $S = l \cdot h$ is the area of the region. We obtain:

$$B_b^{tot}(l) = 2N(l + \frac{S}{l})r + 4Nr^2$$

We derive the expression and obtain:

$$\frac{dB_b^{tot}(l)}{dl} = \frac{l^2 - S}{l^2} \cdot 2Nr$$

The minimum of $B_b^{tot}(l)$ is obtained by setting the derivative equal to zero, which gives:

$l = \sqrt{S}$.

Since the second derivative at $l = \sqrt{S}$ is positive, the minimum communication burden is achieved when the region has a squared shape. This can be obtained when the number of vertical and horizontal partitions[c] of the territory are proportional to the respective sizes of the overall territory, i.e., C_l/C_h=H/L. Of course, the admissible options for the space partitioning also depend on the number of nodes N, since there is the constraint $N = C_l \cdot C_h$.

Figure 11 shows the overall communication burden for a 200x200 territory, partitioned in 64 regions through different combinations of vertical and horizontal partitions. The figure confirms that the minimum communication burden is achieved when regions are square-shaped, i.e., when C_l=C_h=8. Furthermore, it is noticed that in the case that cuts are exclusively horizontal (C_l=64) or exclusively vertical (C_l=1, that is, C_h=64), the communication burden is more than doubled with respect to the optimal case. In Figure 12 we consider a different scenario: a rectangular territory, with L=300 and H=200, partitioned in 96 regions, again with different combinations of horizontal and vertical cuts. In this case the minimum is achieved when the ratio of C_l to C_h is proportional to the ratio of H to L, i.e., C_l=8 and C_h=12, since this way of partitioning the space produces square-shaped regions. It is also noticed that in the case of square-shaped territory the curves of communication burden are symmetrical with respect to the optimal value of C_l (Figure 11), while they are not in the case of rectangular territory (Figure 12).

C. Comparison of linear and bidimensional partitioning

In several cases, the type of space partitioning, linear or bidimensional, is driven by constraints related to the application domain, e.g., the type of data that needs to be processed, the territorial distribution of sensors, etc. In other cases, however, there is more freedom to choose the more convenient territory

[c]It may be useful to notice that, in a toroidal space, the number of horizontal partitions is equal to the number of vertical cuts, and vice versa.

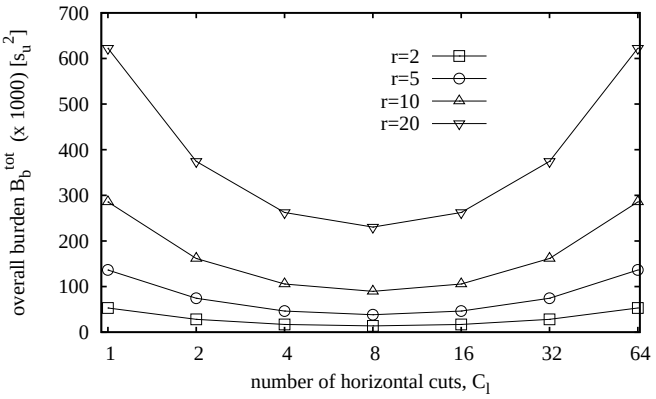

Figure 11. Bidimensional space partitioning: overall burden vs. the number of horizontal cuts C_l, with 64 regions, for different values of the operational radius. $N = 64 = C_l \cdot C_h$. The x axis is in log scale.

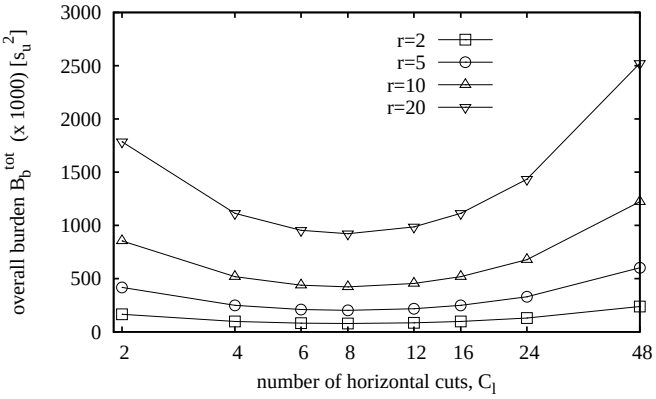

Figure 12. Bidimensional space partitioning: overall burden vs. the number of horizontal cuts C_l, with 96 regions, for different values of the operational radius, in the case of rectangular territory. $N = 96 = C_l \cdot C_h$. The x axis is in log scale.

partitioning. Therefore, it is interesting to compare the communication burdens corresponding to linear and bidimensional space partitioning, to see if one of the two options is more efficient than the other, and in which cases. Specifically, we do the comparison in a scenario where the sizes L and H are given. Moreover, in the case of bidimensional partitioning we consider the case of square-shaped regions, since this proved to be the choice that minimizes the communication burden, as discussed in Section IV-B.

The bidimensional partitioning has an equal or lower overhead when the ratio of expression (3) to expression (2) is equal or lower than one:

$$\frac{B_b^{tot}}{B_l^{tot}} = \frac{2r(l+h) + 4r^2}{2rH} <= 1 \qquad (4)$$

Using the equalities l=h and $C_l/C_h = H/L$, which correspond to square-shaped regions, and considering that the value of the operational radius r is typically much lower than the height of the entire territory H, the inequality (4) is solved

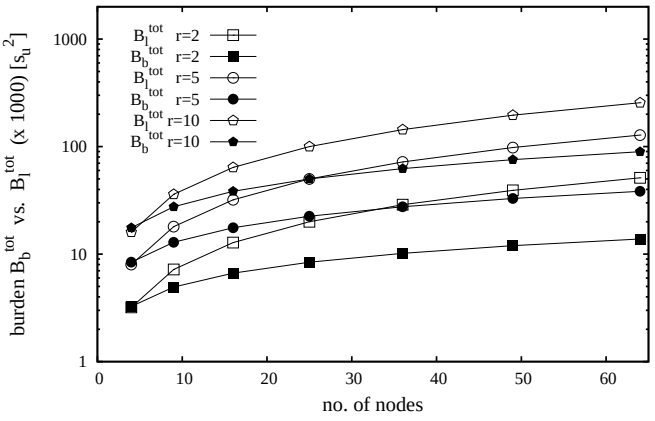

Figure 13. Linear vs. bidimensional space partitioning: overall burden vs. the number of nodes/regions, for different values of the operational radius.

for $C_l >= 2$ and $C_h >= 2 \cdot L/H$. It also corresponds to the inequality $N >= 4 \cdot L/H$. This means, for the case of square-shaped territory (i.e., L=H), that the bidimensional partitioning is a better choice when the number of nodes is equal or larger than 4, and the convenience increases with larger numbers of nodes. This is confirmed in Figure 13, which compares the communication burden of linear and bidimensional partitioning in the case of a 200x200 territory, for different numbers of regions and different values of r. If the territory is rectangular, the minimum number of nodes that makes the bidimensional case more convenient increases as the territory is more and more stretched. For example, this minimum number is equal to N=16 if L=$4H$. Indeed, it is intuitive that the linear partitioning becomes more efficient when the territory extends mostly on one dimension.

V. Conclusion and Future Work

This paper has presented an evaluation of the amount of data that needs to be transferred when executing a wide class of applications on a Cloud infrastructure, i.e., applications that operate on spatial data dislocated in a territory and that exploit the possibility of partitioning such territory to parallelize and speed up the computation. We evaluated different strategies for space partitioning, i.e., mono-dimensional slicing and two types of bidimensional partitioning. We found that the convenience of bidimensional partitioning, in terms of the involved communication burden, increases with the number of adopted computational nodes. We also proved that bidimensional partitioning exhibits its best performances when regions are square-shaped. The outcome of this work will be used by an industrial project that aims to improve the efficiency of electricity and gas distribution in a urban environment. Current work is focusing on the evaluation of the overall execution time in the mentioned scenarios, and on the techniques for dynamically balancing the computational load of different regions.

References

[1] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, 2015.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] Gartner, "Gartner says the internet of things will transform the data center," 2014, March 19. [Online]. Available: http://www.gartner.com/newsroom/id/2684616

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[5] Y. N. Krishnan, C. N. Bhagwat, and A. P. Utpat, "Fog computing-network based cloud computing," in *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*. IEEE, 2015, pp. 250–251.

[6] J. Ekanayake and G. Fox, "High performance parallel computing with clouds and cloud technologies," in *Cloud Computing*. Springer, 2010, pp. 20–38.

[7] R. R. ExpóSito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "Performance analysis of hpc applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218–229, 2013.

[8] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *Open Cirrus Summit (OCS), 2011 Sixth*. IEEE, 2011, pp. 22–26.

[9] E. Roloff, M. Diener, A. Carissimi, and P. O. A. Navaux, "High performance computing in the cloud: Deployment, performance and cost efficiency," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 371–378.

[10] R. Hassani, G. Chavan, and P. Luksch, "Optimization of communication in mpi-based clusters," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2014, pp. 143–149.

[11] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer, "Minimum congestion mapping in a cloud," in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. ACM, 2011, pp. 267–276.

[12] B. Zhang, Z. Qian, W. Huang, X. Li, and S. Lu, "Minimizing communication traffic in data centers with power-aware vm placement," in *6th Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE, 2012, pp. 280–285.

[13] I. Takouna, R. Rojas-Cessa, K. Sachs, and C. Meinel, "Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers," in *Proc. of the 2013 IEEE/ACM Int. Conf. on Utility and Cloud Computing*, 2013, pp. 251–255.

[14] J. Ekanayake, J. Jackson, W. Lu, R. Barga, and A. S. Balkir, "A scalable communication runtime for clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 211–218.

[15] M. Wooldridge, *An introduction to multi-agent systems*. John Wiley & Sons, Ltd., 2002.

[16] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley Longman, 1999.

[17] F. Cicirelli, A. Giordano, and L. Nigro, "Efficient environment management for distributed simulation of large-scale situated multi-agent systems," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 3, pp. 610–632, 2015.

[18] I. Blecic, A. Cecchini, G. A. Trunfio, and E. Verigos, "Urban cellular automata with irregular space of proximities," *Journal of Cellular Automata*, vol. 9, no. 2-3, pp. 241–256, 2014.

[19] M. Pedemonte, S. Nesmachnow, and H. Cancela, "A survey on parallel ant colony optimization," *Applied Soft Computing*, vol. 11, no. 8, pp. 5181 – 5197, 2011.

[20] A. Forestiero and C. Mastroianni, "A swarm algorithm for a self-structured P2P information system," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 681–694, August 2009.

[21] A. Forestiero, C. Mastroianni, and G. Spezzano, "QoS-based dissemination of content in grids," *Future Generation Computer Systems*, vol. 24, no. 3, pp. 235–244, 2008.

[22] F. Cicirelli, A. Forestiero, A. Giordano, and C. Mastroianni, "An approach for scalable parallel execution of ant algorithms," in *International Conference on High Performance Computing & Simulation (HPCS 2014)*, Bologna, Italy, July 2014.

[23] A. Y. Grama, A. Gupta, and V. Kumar, "Isoefficiency: Measuring the scalability of parallel algorithms and architectures," *IEEE Parallel Distrib. Technol.*, vol. 1, no. 3, pp. 12–21, Aug. 1993.