GLOBAL AND LOCAL SYNCHRONIZATION IN PARALLEL SPACE-AWARE APPLICATIONS

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

Rostislav Razumchik
Institute of Informatics Problems
of the FRC CSC RAS, Moscow, Russia,
Peoples' Friendship University of Russia
(RUDN University), Moscow, Russia
Email: rrazumchik@ipiran.ru,
razumchik_rv@rudn.university

KEYWORDS

Synchronization algorithms, parallel computing, performance evaluation

ABSTRACT

Space-aware applications are characterized by an explicit representation of a spatial environment in which some entities live and operate by interacting with each other and with the hosting territory. A relevant space-aware application domain is the so-called urban computing, embracing issues like the simulation and implementation of public transportation systems, traffic management, urban monitoring and control. The execution of such applications is often distributed on parallel computing nodes, which need to cooperate and exchange data among each other, thus raising synchronization issues. In this paper we analyze time-related characteristics of the computational process in a space-aware application in the case when each node does not need *global synchronization* (i.e. synchronization with all other nodes) but requires only *local synchronization* (i.e. synchronization with a subset of neighbor nodes). Performance is evaluated both analytically and numerically. We provide the analytical support to an important conclusion: the mean computation time per step remains finite irrespective of the number of nodes under local synchronization, while under global synchronization it grows unboundedly as the number of nodes increases. In practical scenarios this corresponds to significantly better scalability properties of local synchronization.

INTRODUCTION

The need for parallelizing a computation can be caused by the necessity to increase the efficiency of a very complex application or can be inherent and related to the scenario in which the application is defined. The latter case, on which we focus here, occurs in a large variety of "space-aware applications" (SAAs), for which data and computation are inherently distributed and rely on the explicit representation of a territory, that is, a spatial environment on which data and objects are defined (see Shook et al. (2013)). For example, in an urban environment, data is generated by the users that move in a city, and there can be the need for aggregating and processing the data both at a local level – e.g., at each city neighborhood – and at the a global level, e.g., to derive general knowledge concerning the whole environment.

In such contexts, it is natural to parallelize the computation on distributed nodes and to perform the partitioning by utilizing the topological properties of the territory itself. Specifically, different regions of the territory can be assigned to different computing nodes which can process local data in parallel. Geology, biology, hydrology, social sciences, logistics and transportation, smart electrical grids, are significant examples of application fields strongly related to SAAs (see Cicirelli et al. (2016); Gong et al. (2013); Tang et al. (2011)). Another one is the urban-computing field mentioned above, for which data contains information regarding the mobility of people or vehicles, air quality, safety issues, water/electricity consumptions, etc., and can be profitably used to improve urban services and environments (see Zheng et al. (2014); Blecic et al. (2014)). Two more application fields that are emerging recently are the "Internet of Things" (IoT) (see Atzori et al. (2010); Lee and Lee (2015)) and some new distributed forms of Cloud Computing, sometimes referred to as Fog Computing or Edge Computing (see Krishnan et al. (2015); Hu et al. (2017)), where the computation is brought closer to the user's end and/or where the data is generated.

In general, space-aware applications are not "embarrassingly parallel" (Ekanayake and Fox, 2010), i.e., computation at the single nodes cannot be performed in isolation because parallel tasks need to exchange data during computation. This means that the computation performed by different nodes needs to be synchronized (Fujimoto, 2000), i.e., at certain time instants, one node must wait for the data coming from other nodes before proceeding to the next piece of computation. In this paper, for the sake of simplicity, we focus on the very common case of step-based computation (i.e., the computation is organized in work units which we call "steps"), and synchronization occurs at the end of each step.

Given the growing interest in space-aware applica-

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

tions, there is a strong need for methodological approaches that help assessing the performance of their parallel execution. In particular, it is a well-established empirical fact that the methodology adopted for synchronization has a notable impact on the performance. With the widely adopted all–to–all synchronization approach, henceforth also referred to as *global*, a computation step can be executed only as soon as all the nodes have completed the previous step. However, in many contexts this requirement can be relaxed, and a node can proceed to the next step after synchronizing with a limited number of nodes, e.g., those which are assigned to adjacent portions of the territory in a urban computing scenario. This type of synchronization is henceforth referred to as *local synchronization*.

One example of an application that can profitably adopt local synchronization is the computation of frequent mobility patterns (Yuan et al., 2013), i.e., the most common routes that are followed by vehicles, as these patterns can be extracted by concatenating the local patterns discovered in different city districts (Harri et al., 2009). The opportunity emerges of synchronizing the computation only among a limited number of parallel nodes, without the need for a central coordinator node. The computation at one node can proceed after being notified about the patterns discovered in neighbor nodes, and can then concatenate these patterns, thus allowing mobility patterns to be available much more rapidly than in the scenario where the computation is synchronized globally and data is delivered to a central node. The patterns regarding the whole territory are achieved by progressively extending the area covered by local patterns.

In Mastroianni et al. (2017) it was shown that local synchronization performs better than global synchronization in terms of computation efficiency. However, there is the need for building a theoretical foundation that is able to accurately predict the performance, and also to analyze the scalability properties. The scalability issue arises when the number of parallel nodes increases, either because the analyzed territory is extended (for example, the city area for the mobility pattern application mentioned before) or because the same area is divided into a larger number of regions in order to improve the accuracy of the computation. An analytical study is essential to tackle important engineering issues, such us: what is the number of parallel nodes needed to execute the computation in a given time interval? what is the impact of synchronization degree, i.e., the number of nodes with which a single node must synchronize?

In this paper, we provide an analytical representation of the model and its key ingredients: the overall execution time and the time to perform a single step when the synchronization overhead is taken into account. In particular, we prove that in the general case, the average time to perform a computation step on each node converges under local synchronization, i.e., it is bounded when the number of nodes increases, while under global synchronization it grows unboundedly. Therefore, the adoption of local synchronization is of utmost importance when the computational load or the involved scenario requires the use of a large number of nodes.

The paper is organized as follows. The next section describes the local synchronization model and provides an analytical formulation for the execution time. Then we assess the performance of local and global synchronization: at first analytically by using the extreme value theory and the max-plus algebra, then numerically by simulation. In conclusion, we summarize the work and indicate some interesting research avenues for future work.

LOCAL SYNCHRONIZATION MODEL

A natural way to optimize the execution of algorithms working on spatial data is to partition the territory and use this partitioning to decompose and parallelize the computation. The idea is to individuate a number of *regions* of the territory, and assign each region, along with the contained entities, to a *computing node* that will be in charge of performing the computation pertaining to that portion of the territory. 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 *one-dimensional* or a *bidimensional* schema, as shown in Figure 1.

Let us denote by N the number of nodes, by $l_i(k)$ the time needed by node i, $1 \leq i \leq N$, to execute the local computation at the step k, and by $T_i(k)$ the time elapsed from the beginning of the computation at node i (i.e., start of the step 1) until the end of the step k.

It is important to define how the computing nodes synchronize with each other. In many parallel/distributed systems, as reported in the current literature, synchronization is global or, in other words, *all–to–all*, i.e., it is performed by constraining each node to start the execution for a given step only when all the nodes have finished their execution of the step before. The left part of Figure 2 shows an example of the dynamics of a system composed of seven nodes, for two consecutive steps. It is seen in the figure that, at each step, all the nodes must wait for the slowest one before advancing to the next step. In the figure, node 5 is the slowest node at step 1 while node 3 is the slowest at step 2.

For many SAAs scenarios, global synchronizations is not required. Indeed, as mentioned in the introduction section, it can be that a node needs to communicate and synchronize only with a set of neighbor nodes. Figure 3 shows the loop executed by each computing node, at each step, when adopting such local synchronization. The loop includes three phases. First, the node executes the local computation, i.e., the computation related to the specific region for the current step. Afterwards, the node sends data to its neighbor nodes. Finally, the node waits for the analogous data coming from its neighbors, i.e., the nodes managing the neighbor regions.

Resuming the execution advancement shown in the left part of Figure 2, corresponding to the case of global syn-

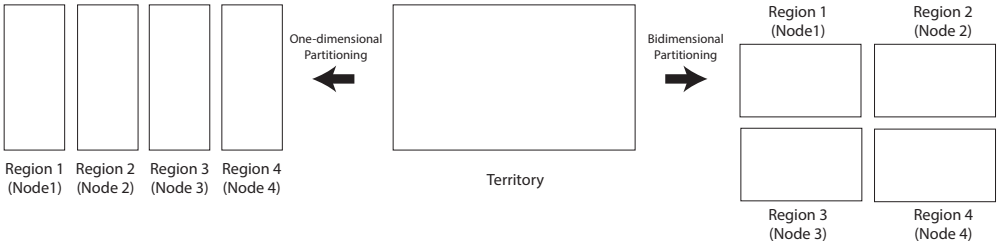

Figure 1: A territory partitioned into regions which are associated with parallel computing nodes. Two alternative types of partitioning are shown, one-dimensional and bidimensional.

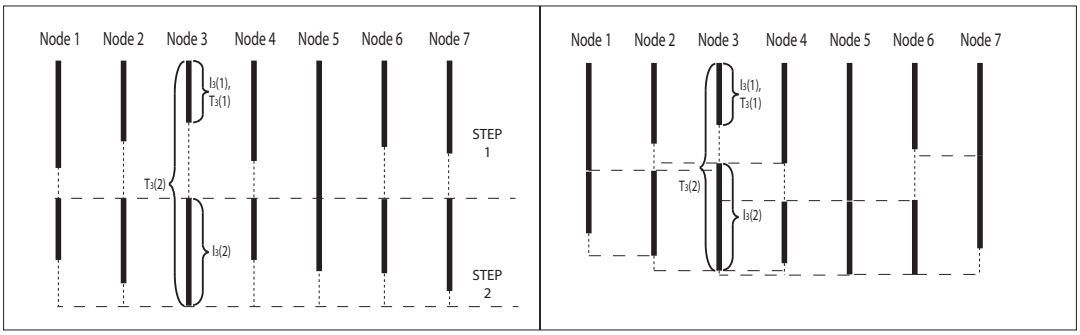

Figure 2: Dynamics of seven nodes for two steps using global synchronization (left) and local synchronization (right). The solid vertical lines represent the execution times, the dashed vertical lines are the waiting times and the horizontal dashed lines represent the synchronization points.

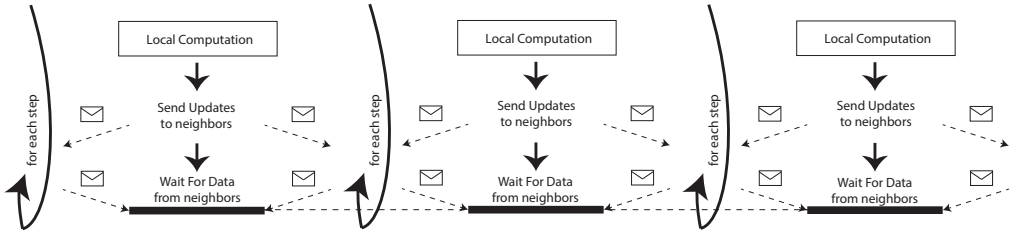

Figure 3: Execution loop under local synchronization.

chronization, we take the same local computation times, $l_i(k)$, and in the right part of Figure 2 we show the corresponding execution advancement when adopting the local synchronization. It can be seen that the times $T_i(k)$ tend to be shorter when compared to the case of global synchronization.

Under local synchronization, $T_i(k)$ are determined from the following recursive formula:

$$T_i(k+1) = \max\left(T_i(k), T_{i-1}(k), T_{i+1}(k)\right) + \\ + \; l_i(k+1), \; 1 \le i \le N, \quad (1)$$

where $T_0(k) = T_{N+1}(k) = 0$.

In (1) we have implicitly assumed that the time for transmitting the data between the neighbor nodes is negligible. Let $c_{i,j}(k)$ be the communication time needed for transmitting the data from the node i to the node j at the end of step k. When the communication time is not negligible, the recursive formula (1) is transformed to:

$$T_i(k+1) = \max\left(T_i(k), T_{i-1}(k) + c_{i-1,i}(k), \right. \\ \left. T_{i+1}(k) + c_{i+1,i}(k)\right) + \; l_i(k+1), \; 1 \le i \le N. \quad (2)$$

In the case of bidimensional partitioning, using a grid with R rows and C column, and $N = R \cdot C$, let us call $i_{r,c}$ the node located in row r, $1 \le r \le R$, and column c, $1 \le c \le C$. For example, in the right part of Figure 1, $i_{1,1}$ is Node 1 and $i_{1,2}$ is Node 2. Accordingly, $l_{r,c}(k)$ and $T_{r,c}(k)$ are, respectively, the local computation time and the time elapsed from the beginning of the computation to the end of the step k at the node $i_{r,c}$, while $c_{r,c,r',c'}(k)$ is the time for transmitting data from the node $i_{r,c}$ to the node $i_{r',c'}$ at the end of step k. In the bidimensional case, $T_{r,c}(k)$ are computed from the recursion:

$$T_{r,c}(k+1) = \max(T_{r,c}(k), T_{r-1,c-1}(k) + c_{r-1,c-1,r,c}(k), \\ T_{r-1,c}(k) + c_{r-1,c,r,c}(k), T_{r-1,c+1}(k) + c_{r-1,c+1,r,c}(k), \\ T_{r,c-1}(k) + c_{r,c-1,r,c}(k), T_{r,c+1}(k) + c_{r,c+1,r,c}(k), \\ T_{r+1,c-1}(k) + c_{r+1,c-1,r,c}(k), T_{r+1,c}(k) + c_{r+1,c,r,c}(k), \\ T_{r+1,c+1}(k) + c_{r+1,c+1,r,c}(k)) + \; l_{r,c}(k+1) \\ 1 \le r \le R, \; 1 \le c \le C, \quad (3)$$

where $T_{0,c}(k) = T_{R+1,c}(k) = T_{r,0}(k) = T_{r,C+1}(k) = 0$.

Despite the relative simplicity of the systems of equations (2) and (3), which govern the behavior of the synchronization model, it turns out to be very hard to come up with the analytic analysis of its performance characteristics. In the next section we dwell on only one aspect of this problem: analysis of the mean computation time per step.

PERFORMANCE OF GLOBAL AND LOCAL SYNCHRONIZATION

In what follows we give the analytic support to the following conclusion: global synchronization leads to unbounded mean computation time per step as the number of nodes increases, whereas the local synchronization guarantees that the mean computation time per step remains finite irrespective of the number of nodes.

In what follows, for the sake of ease of exposition, we dwell on the simple case of the model: one-dimensional partitioning and negligible communication times. In addition, we assume that the computation times $l_i(k)$ depend only on the nodes but do not depend on the step number and that $l_i(k) = l_i$, $1 \le i \le N$, are i.i.d. random variables. In the case of global synchronization we have

$$T_i(k+1) = (k+1)\max(l_1, \dots, l_N),\ k \ge 0, \qquad (4)$$

and in the case of local synchronization we have (1), which is readily reduced to

$$T_i(k+1) = \max(T_i(k), T_{i-1}(k), T_{i+1}(k)) + \\ + l_i,\ k \ge 0,\ 1 \le i \le N. \qquad (5)$$

Note that the random sequence $\{\vec{T}(k)/k,\ k \ge 1\}$, where $\{\vec{T}(k) = (T_1(k), \dots T_N(k))\}$ and $T_i(k)$ are defined by (1) (and, of course, (4)), falls into the framework of stochastic equations as described in Borovkov (1979). Using the results from Borovkov (1979) it can be shown that when N is finite and $\{\vec{l}(k) = (l_1(k), \dots, l_N(k)),\ k \ge 1\}$ are independent, the sequence $\{\vec{T}(k)/k,\ k \ge 1\}$ is ergodic and stable.

Analysis of global synchronization

Let us start with the analysis of (4). The conclusion about the behavior of the global synchronization case follows from the well-known results for the order statistics and extreme value theory (see, for example, David and Nagaraja (2003); Ang and Tang (1984); Madala and Sinclair (1991)). If the positive random variable l_i has any continuous distribution with the support[1] on a semi-infinite interval, then as the number of nodes N grows, the mean computation time per step $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$ grows as well and in the limit as $N \to \infty$ we have that

$$\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k)) = \mathbf{E}(\max(l_1, \dots, l_N)) \to \infty.$$

For example, if l_i are i.i.d random variables distributed exponentially with mean μ, then for sufficiently large N we have[2] $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k)) \approx \mu\,(\ln N + 0.5772)$.

Such nice expressions for extreme values are not available for all distributions. Yet for quite a large class of distributions (those having pure-exponential and non-pure exponential tails like gamma distribution) the general expressions for $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$ can be found in Whitt et al. (2007).

So in the case of global synchronization, if the computation times are random with unbounded support, then as the number of nodes N increases, on average, we wait longer and longer in order to make the next computation step. The computation of the cumulative distribution function (c.d.f.) of $\frac{1}{k}\mathbf{E}(T_i(k))$ is straightforward when l_i are i.i.d. It should also be noticed that deeper insight into the global synchronization can be gained by looking at the relation between the global synchronization model and the two types of queueing models: split-merge queues[3] (see, for example, Altiok and Perros (1986)) and faucet queues as described in Lebedev (2003, 2004).

Analysis of local synchronization

Once we give up the global synchronization and allow the node to proceed to the next computation step once a finite number of neighbors finish their computations, the conclusion is changed: the mean computation time per step $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$ becomes finite irrespective of the number of nodes N. In the following we consider the case of synchronization with two neighbors[4], which is described by (5).

The model (5) falls into the general framework of discrete event dynamic systems and it is convenient to describe its evolution in terms of the max-plus algebra (Heidergott et al., 2006). As soon as it is done, one can use the well-known results of the max-plus theory to study the values of $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$.

Define the following "(max,+)" notations:

$$\forall x, y, \in \mathbb{R} \cup \{-\infty\},\ x \oplus y = \max(x, y),\ x \otimes y = x + y.$$

Define also the $N \times N$ matrix $\mathbf{T}(k) = [\mathbf{T}(k)]_{ij}$, $1 \le i, j \le N$. With this notation, remembering that $\vec{T}(k) = (T_1(k), \dots T_N(k))$, equation (5) can be rewritten as

$$\vec{T}^{\,T}(k+1) = \mathbf{T}(k) \otimes \vec{T}^{\,T}(k),\ k \ge 0, \qquad (6)$$

where $\cdot^T$ stands for transpose, the matrix-vector product is defined by $[\mathbf{T}(k) \otimes \vec{T}^{\,T}(k)]_i = \max_{1\le j\le N}([\mathbf{T}(k)]_{ij} + T_j(k))$

[1] In the case that l_i are independent (not necessarily identically distributed) random variables with a continuous distribution having support on a bounded interval, the mean computation time $\lim_{k\to\infty} \mathbf{E}(T_i(k))/k$ is always a constant, irrespective of the number of nodes N.

[2] It is well-known that the right part is the approximation of the expected maximum of N i.i.d. random variables with exponential distribution equal to $\mu \sum_{i=1}^{N} i^{-1}$, with $\sum_{i=1}^{N} i^{-1}$ being the N^{th} harmonic number.

[3] The sojourn time of the k^{th} customer in a split-merge queue in the light-traffic regime is equal to $\frac{1}{k}\mathbf{E}(T_i(k))$.

[4] But the analysis and conclusions remain valid also in the case of more than two neighbours and in the case of (at least simple) bidimensional partitioning.

and the matrix $\mathbf{T}(k)$ is defined by

$$\mathbf{T}(k) = \begin{pmatrix} l_1 & l_1 & -\infty & -\infty & \dots & -\infty & -\infty \\ l_2 & l_2 & l_2 & -\infty & \dots & -\infty & -\infty \\ -\infty & l_3 & l_3 & l_3 & \dots & -\infty & -\infty \\ -\infty & -\infty & l_4 & l_4 & \dots & -\infty & -\infty \\ \vdots & \vdots & \vdots & \vdots & \ddots & \dots & \ddots \\ -\infty & -\infty & -\infty & -\infty & \dots & l_{N-1} & l_{N-1} \\ -\infty & -\infty & -\infty & -\infty & \dots & l_N & l_N \end{pmatrix}.$$

Here the initial condition $\vec{T}^T(0)$ is simply the column-vector of zeros. Now we can make use of the well-known asymptotic results from the max-plus theory (see, for example, Baccelli and Konstantopoulos (1992)). The matrix $\mathbf{T}(k)$ has at least one finite entry on each row, which is the necessary and sufficient condition for $T_j(k)$ to be finite. From (Baccelli and Konstantopoulos, 1992, Lemma 6.1) we find[5] that there exists $\gamma > 0$ such that $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k)) = \gamma$, $1 \le i \le N$. In the case when l_i are stochastically bounded by a single variable (say L), having moment generating function (say $L(s)$), the upper bound for the value of γ is (see (Baccelli and Konstantopoulos, 1992, Proposition 6.2)):

$$\gamma \le \inf\{x > \mathbf{E}(L) \text{ such that } M(x) > \ln 3\}, \qquad (7)$$

where $M(x) = \sup_{\theta\in\mathbb{R}}(\theta x - \ln L(\theta))$ and $\mathbf{E}(L) = L'(0)$.

What this result tells us is that in the case of local synchronization the mean computation time remains finite for any number of nodes N. We could also proceed with the establishing of the upper bound in (7) right from (5) without resorting to the results of the max-plus algebra. Indeed, assuming, as above, that all l_i are stochastically bounded by L with the moment generating function $L(s)$, we can apply stochastic ordering techniques to find the following upper bound for $\lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$: $(\ln 3 + \ln L(s))/s$, where s is the solution to the equation $L(s) = 3$.

The exact computation of γ in the considered model is difficult and we are unaware of any well-established procedure (either in max-plus world or conventional probability theory) to perform such computation. On the contrast to the global synchronization model, where the exact value of γ can be easily written out, the main problem in the local synchronization is the inter-dependence between the values of $T_i(k)$ and $T_j(k)$, $i \neq j$. If we drop this dependence by assuming that $\{T_i(k),\ 1 \le i \le N\}$ are independent for each k, the computations of the c.d.f. of $T_i(k)$ are possible but they may lead to highly underestimated or overestimated values of γ. If we keep the dependence structure, then at each step k we have to perform the computations of the N-dimensional c.d.f. of the vector $\vec{T}(k)$, which already for small values of k and N become infeasible. At last, we note that the structure of (5) suggests that for the construction of the N-dimensional c.d.f. it is appealing to apply the dependence trees technique (see Chow and Liu (1968)), which takes into account order dependence relationships between the random variables.

[5] Baccelli and Konstantopoulos (1992) gives even stronger results of convergence of $\lim_{k\to\infty} \frac{1}{k}(T_i(k))$ to the same γ with probability 1.

Numerical results

In the following we report some numerical results both for global and local synchronizations under: one-dimensional partitioning, negligible communication times and i.i.d. computation times l_i. We have used Matlab to simulate the computational behavior modeled by (4) and (5) with different number of nodes N. We have also considered the extended local synchronization scenario in which a node synchronizes with more than two neighbors (the number of neighbors with which a node synchronizes on each of the two sides (left and right) is referred to as N_b). For l_i we have considered[6] exponential and hyper-gamma distributions. The performance is assessed by computing[7] the mean computation time per step $T_{step} = \lim_{k\to\infty} \frac{1}{k}\mathbf{E}(T_i(k))$.

First, we consider the case in which l_i has an exponential distribution with the mean equal to 6.185, for a fair comparison with the hyper-gamma distribution discussed later. Figure 4 shows the values of T_{step} versus the number of nodes N in the case of global and local synchronization with different values of N_b. The figure also reports the theoretical value for the case of global synchronization, which corresponds to the N^{th} harmonic number. The experimental values are consistent with the theoretical bounds discussed in the previous section, and it can be seen that the use of local synchronization allows T_{step} to be notably reduced with respect to global synchronization, even when the number of neighbors N_b increases. In Figure 5 we focus on the "exponential" scenario with N_b=1 and report the bound obtained with the

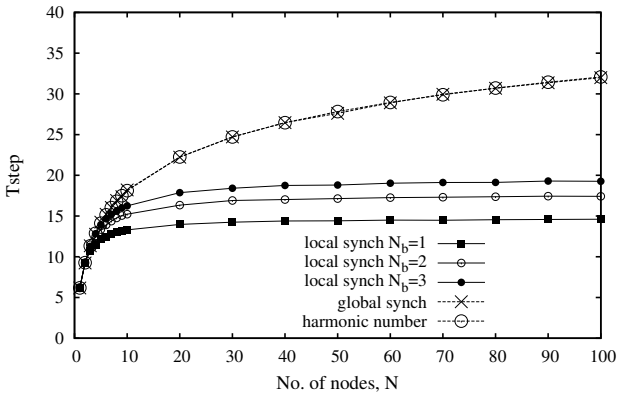

Figure 4: Values of T_{step} as function of N under global and local synchronizations with different values of N_b. The local computation times have exponential distribution with mean 6.185.

[6] Our choice is motivated by (Lublin and Feitelson, 2003), where hyper-gamma distribution is shown to be well-suited workload model in parallel computing systems.

[7] In order to compute each time the steady state value of T_{step} we used a single simulation run with $k = 10000$ and the batch-means method.

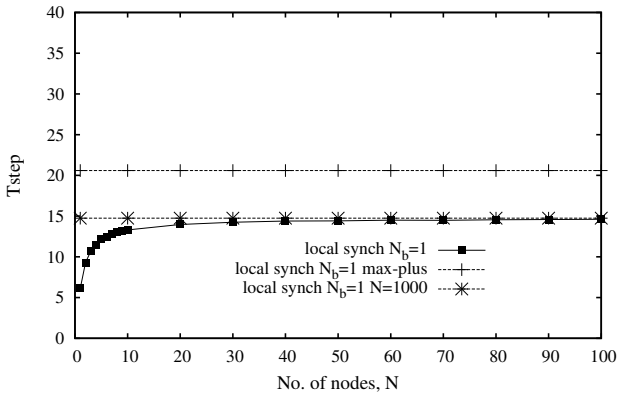

Figure 5: Values of T_{step} as function of N under local synchronization and N_b=1. The local computation times have exponential distribution with mean 6.185. The plot shows the bound obtained with max-plus algebra and the numerical value with N=1000.

max-plus algebra. We also show the value for $N = 1000$ nodes. From the figure it can be seen that the exact numerical bound is already reached when the number of nodes is 100.

Figure 6 shows the results that are analogous to those of Figure 4, this time assuming that the local computation times have hyper-gamma distribution with the pdf

$$p(x) = p\frac{1}{\Gamma(\alpha_1)}\beta_1^{\alpha_1} x^{\alpha_1-1} x^{-\beta_1 x} + (1-p)\frac{1}{\Gamma(\alpha_2)}\beta_2^{\alpha_2} x^{\alpha_2-1} x^{-\beta_2 x},$$

and the parameters p, α_i, β_i taken from (Lublin and Feitelson, 2003, Table 2):

$$p = 0.55, \alpha_1 = 6, \beta_1 = 1.51, \alpha_2 = 68.5, \beta_2 = 7.692.$$

It is seen that the advantage of local synchronization is larger with the exponential distribution due to its larger variance. This is a general result that we have also found in other experiments not shown here.

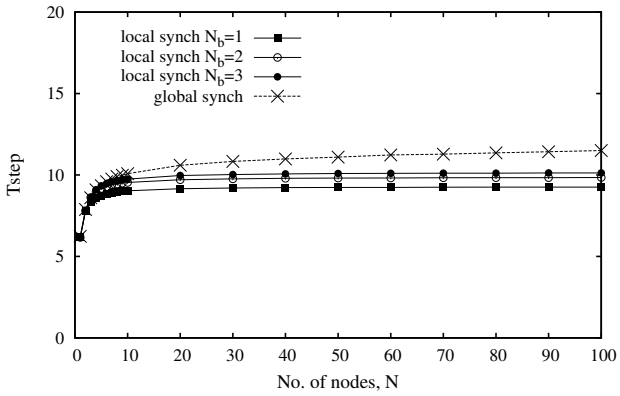

Figure 6: Values of T_{step} as function of N under global and local synchronizations with different values of N_b. The local computation times have hyper-gamma distribution, with mean equal to 6.185 and variance equal to 8.011.

CONCLUSION

In this paper two different synchronization strategies are being compared, namely local and global, for the execution of distributed space-aware applications. We have provided an analytical support, based on the max-plus theory, to conclusions about the finiteness of the computation time per step under local synchronization in general, and its unboundedness (for infinite number of nodes) under global synchronization. In practical scenarios, this corresponds to a much better scalability behavior of local synchronization, as confirmed by numerical experiments. We deem that this result can be profitably considered when designing a real computational infrastructure, for example for the support of urban computing applications. The problem of exact computation of moments of computation time per step in the asymptotic case is still open and requires further study. Also, the benefits deriving from local synchronization need to be better assessed in cases when communication times are not negligible and the computation load is not equally partitioned among the nodes and/or varies with time.

REFERENCES

Altiok, T. and Perros, H. (1986). Open networks of queues with blocking: Split and merge configurations. *IIE Transactions*, 18(3):251–261.

Ang, A.-S. and Tang, W. (1984). *robability Concepts in Engineering Planning and Design Vol. II*. Rainbow Bridge.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

Baccelli, F. and Konstantopoulos, P. (1992). Estimates of cycle times in stochastic petri nets. In *Applied Stochastic Analysis*, pages 1–20. Springer Berlin Heidelberg.

Blecic, I., Cecchini, A., Trunfio, G. A., and Verigos, E. (2014). Urban cellular automata with irregular space of proximities. *Journal of Cellular Automata*, 9(2-3):241–256.

Borovkov, A. A. (1979). Ergodicity and stability theorems for a class of stochastic equations and their applications. *Theory of Probability & Its Applications*, 23(2):227–247.

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Cicirelli, F., Forestiero, A., Giordano, A., and Mastroianni, C. (2016). Transparent and efficient parallelization of swarm algorithms. *ACM Trans. Auton. Adapt. Syst.*, 11(2):14:1–14:26.

David, H. A. and Nagaraja, H. N. (2003). *Order Statistics, Third Edition*. John Wiley.

Ekanayake, J. and Fox, G. (2010). High performance parallel computing with clouds and cloud technologies. In *Cloud Computing*, pages 20–38. Springer.

Fujimoto, R. (2000). *Parallel and distributed simulation systems*. John Wiley.

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):1152–1170.

Harri, J., Filali, F., and Bonnet, C. (2009). Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4):19–41.

Heidergott, B., Olsder, G. J., and van der Woude, J. (2006). *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press.

Hu, P., Dhelim, S., Ning, H., and Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98:27 – 42.

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

Lebedev, A. V. (2003). The gated infinite-server queue with unbounded service times and heavy traffic. *Problems of Information Transmission*, 39(3):309–316.

Lebedev, A. V. (2004). Gated infinite-server queue with heavy traffic and power tail. *Problems of Information Transmission*, 40(3):237–242.

Lee, I. and Lee, K. (2015). The internet of things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431 – 440.

Lublin, U. and Feitelson, D. G. (2003). The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105 – 1122.

Madala, S. and Sinclair, J. B. (1991). Performance of synchronous parallel algorithms with regular structures. *IEEE Trans. on Parallel and Distributed Systems*, 2(1):105–116.

Mastroianni, C., Cesario, E., and Giordano, A. (2017). Efficient and scalable execution of smart city parallel applications. *Concurrency and Computation: Practice and Experience*. Early view, http://dx.doi.org/10.1002/cpe.4258.

Shook, E., Wang, S., and Tang, W. (2013). A communication-aware framework for parallel spatially explicit agent-based models. *International Journal of Geographical Information Science*, 27(11):2160–2181.

Tang, W., Bennett, D. A., and Wang, S. (2011). A parallel agent-based model of land use opinions. *Journal of Land Use Science*, 6(2–3):121–135.

Whitt, W., Crow, C., and Goldberg, D. (2007). Two-moment approximations for maxima. *Operations Research*, 55(3):532–548.

Yuan, J., Zheng, Y., Xie, X., and Sun, G. (2013). T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):220–232.

Zheng, Y., Capra, L., Wolfson, O., and Yang, H. (2014). Urban computing: Concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol.*, 5(3):1–55.

AUTHOR BIOGRAPHIES

Franco Cicirelli Ph.D, is a researcher at the ICAR-CNR Institute, Italy, since 2015. He earned a Ph.D. in System Engineering and Computer Science at the University of Calabria (Italy). He was a researcher fellow at the University of Calabria (Italy) from 2006 to 2015. Research topics include agent-based systems, distributed simulation, parallel and distributed systems, real-time systems, workflow management systems, Internet of Things and cyber-physical systems.

Agostino Forestiero is a researcher at ICAR-CNR, Italy, since 2010. He received his Laurea degree and his Ph.D. degree in Computer Engineering from the University of Calabria, Italy, in 2002 and 2007. He co-authored over 50 papers published in international journals, among which IEEE/ACM TON, IEEE TEVC and ACM TAAS, and conference proceedings. His areas of interest are distributed systems, Cloud Computing, P2P, swarm intelligence, multi-agent systems and cyber-physical systems.

Andrea Giordano is a researcher at ICAR-CNR, Italy, since 2011. He earned a Masters degree in Computer Engineering and a Ph.D in System Engineering and Computer Science at the University of Calabria, Italy. His research work focuses on agent-based systems, parallel and distributed systems, swarm intelligence, distributed simulation and cyber-physical systems.

Carlo Mastroianni is a senior researcher at ICAR-CNR, Italy. He received his Laurea degree and his Ph.D. degree in Computer Engineering from the University of Calabria, Italy, in 1995 and 1999, respectively. He authored over 100 papers published in international journals, among which IEEE/ACM TON, IEEE TCC, IEEE TEVC and ACM TAAS, and conference proceedings. His research focuses on Cloud Computing, P2P, bio-inspired algorithms, multi-agent systems.

Rostislav Razumchik received his Ph.D. degree in Physics and Mathematics in 2011. Since then, he has worked as a leading research fellow at Institute of Informatics Problems of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (FRC CSC RAS). Currently he also holds the associate professor position at Peoples' Friendship University of Russia (RUDN University). His current research activities are focused on queueing theory and its applications for the evaluation of stochastic systems.