Received: 16 November 2016 | Revised: 3 May 2017 | Accepted: 23 June 2017
DOI: 10.1002/cpe.4258

WILEY

SPECIAL ISSUE PAPER

Efficient and scalable execution of smart city parallel applications

Carlo Mastroianni | Eugenio Cesario | Andrea Giordano

ICAR-CNR, Via P. Bucci, 87036 Rende (CS), Italy

Correspondence
Carlo Mastroianni, ICAR-CNR, Via P. Bucci, 87036 Rende (CS), Italy.
Email: mastroianni@icar.cnr.it

Funding information
MIUR, Grant/Award Number: PON03PE_00050_2

Summary

Recent research efforts in the field of urban computing aim to develop innovative services for citizens through the application of ubiquitous and pervasive computing paradigms in urban spaces. Smart city applications need to cope with a large number of involved users and devices. Since data and objects are strictly related to the territory on which they are defined and used, it is preferable, when possible, to perform computation locally through the adoption of dispersed computing nodes such as CPU-equipped sensors. In this context, the computation related to smart city applications can be profitably and efficiently parallelized by partitioning the territory into regions and assigning the computation related to each single region to a local node. Nevertheless, the adoption of parallel computing models poses several communication and synchronization issues, especially when the number of nodes is large and the time constraints of applications are compelling. This paper presents and analyzes a parallel computing model for smart city applications in which each node needs to exchange information only with a subset of neighbor nodes, allowing the synchronization overhead to be significantly reduced. As sample application, we consider the analysis and prediction of internet traffic generated by vehicle and pedestrian devices moving on a smart avenue equipped with distributed computing nodes. This work offers a detailed performance evaluation in a number of scenarios, including uniform and nonuniform user distribution and different types of user mobility behavior. The results show that the presented computation model offers notable advantages in terms of computation efficiency and speedup, with respect to a classical all–to-all synchronization paradigm, in which the nodes need to coordinate with a central entity.

KEYWORDS
parallel computation, smart city, synchronization, urban computing

1 | INTRODUCTION

The widespread diffusion of sensing technologies and large-scale computing infrastructures has enabled the collection of big and heterogeneous data that are daily produced in urban spaces. Such data are pertaining to the mobility of people or vehicles, air quality, safety issues, water/electricity consumptions, etc, and represent useful resources to improve urban services and environments. This is stimulating several research efforts in the field of *urban computing*, an interdisciplinary scientific field that pertains to the study and application of computing technology in urban environments.[1-3] In particular, the *smart city* paradigm aims to plan and design future urban territories in a more efficient way. Smart cities rely on the adoption of IT technologies, sensors, Web cams, databases, IoT systems, ubiquitous devices, wireless networks, and all those frameworks that are used for sensing cities and territories, for collecting data and for acting on the basis of the applications logic. In order to process the big amount of data being produced, smart cities are supposed to use multiple hardware and software technologies including parallel system architectures and parallel programming, eg, low-level programming models and skeletal parallel programming.[4]

Due to the specific nature of smart city applications, data and objects are strictly related to the space or territory on which they are defined and used.[5,6] Indeed, environmental information extracted from sensors, data inherent to the neighborhoods and residential units in a city, information about traffic and mobility, etc, are specifically descriptive of the city area where they have been gathered and stored. For such a reason, the computation related to smart city applications can be profitably partitioned and parallelized by assigning different areas of the city to different computing entities, for example servers or smart sensors, in order to perform the computation as close as possible to data sources.

A suitable technological solution to tackle mobility and geo-distribution of data, embrace location awareness and ensure low latency, can be provided by a variant of cloud computing, referred to as *fog computing*,[7,8] which relies on a number of distributed cloud facilities located close to data sources, ie, a cloud close to the ground. The fog computing paradigm allows the computational load to be partitioned among the distributed network of devices and the more powerful centralized servers. A proper distribution of computation can be more efficient and practical than the classical approach in which data are collected from sensors and conveyed to centralized units where all the computation is performed, for two main reasons: (1) the amount of data to be transmitted can be huge, for example, in video applications, possibly leading to bandwidth problems. This issue can be alleviated by performing part of the computation locally and (2) in several applications it is mandatory to obtain computational results in a short time interval or even in real time. This can be unfeasible if all the computation is allocated to the centralized/cloud component. In such an environment, it is essential to establish the proper trade-off between local/distributed and global/centralized computation.[9,10] For example, distributed devices could be devoted to achieve real-time results that are needed locally, while centralized servers could focus on intense offline computation aimed at extracting long-term patterns. Nevertheless, such large-scale integration and the adoption of parallel computing models pose several communication and synchronization issues, especially when the number of nodes is very high and the time constraints of applications are compelling.

This paper extends the preliminary study presented in Mastroianni et al[11] and is a pioneering contribution to the mentioned research avenues in that it offers a preliminary performance evaluation of a fog computing infrastructure that enables the execution of efficient parallel applications in a smart city scenario. As a sample parallel application, in this work we focus on the analysis and prediction of internet traffic[12-14] generated by vehicle and pedestrian devices moving across a "smart avenue," even though our study can be extended to a variety of applications. In particular, we consider smart city applications requiring that the computation regarding a specific region of the city is performed using the information received from a subset of neighbor regions. This scenario opens the opportunity of synchronizing the computation only among a limited number of parallel nodes, without the need for a central coordinator node. Thus, our main goal is to analyze the advantage deriving from such a *local synchronization* paradigm, when compared to the *all–to–all synchronization* approach, where all the nodes need to synchronize before proceeding to the next computation step.

In addition, we also investigate how the efficiency and speedup obtained with the two synchronization paradigms are influenced by (1) different distributions of mobile users along the avenue and (2) different probability distributions of the computational times experienced at the single nodes. We analyze how the tuning of system parameters—specifically, the overall number of computing nodes and the number of neighbor nodes that need to synchronize among each other—has effect on the trade-off between computational efficiency and accuracy. We consider the case of nonuniform distribution of mobile users, in both static scenarios (users distribution is steady) and dynamic scenarios (a large number of users simultaneously move in the same direction along the avenue). Moreover, we present a Petri net that we use to model and analyze the smart avenue and can be easily adapted to different scenarios, for example, a bidimensional territory. Indeed, the Petri net formalism is particularly suitable to capture and analyze the performance related to synchronization.

The rest of the paper is organized as follows. Section 2 describes the smart avenue scenario considered for this work. Section 3 illustrates how the synchronization among neighbor regions can be modeled through a Petri net. Section 4 reports performance results, in terms of computation time, speedup, and accuracy of the computation, when varying the number of parallel nodes, the synchronization degree and the distribution and mobility behavior of users, and summarizes the main outcomes that can be derived from the results. Section 5 outlines related work concerning the parallel execution of smart city applications, with a particular focus on the analysis of Internet traffic in IoT and cloud/fog-based urban computing environments. Finally, Section 6 concludes the paper and plans future research works.

2 | SMART AVENUE SCENARIO

The smart city application used as a test case in this work is the analysis of the Internet traffic generated by the devices located and moving over a city avenue. This choice allows us to start with a mono-dimensional scenario, as it is simpler to model and the related results are easier to analyze. Afterward, the analysis can be naturally extended to a two- or three-dimensional scenario. The smart avenue model consists in a large road on which pedestrians and vehicles generate Internet traffic to use classical audio/video applications, for example, social applications or navigators. In addition, as envisioned by the Cloud of Things paradigm, in particular by the vehicular cloud scenario,[15] smart devices can offer their computing and storage capabilities to perform computations in combination with the facilities of a fixed cloud infrastructure. The goal of the smart avenue application is to analyze and predict the amount and characteristics of the data network traffic in a future interval of time, starting from the past behavior of mobile devices.[12-14] In this context, past behavior concerns both the usage of internet applications and the mobility behavior of the users.

The prediction of Internet traffic can be performed both at local and global level. At local level, each node can communicate with its neighbors and discover traffic pattern models within its competence area. The results can be used, for example, to generate real-time alerts regarding traffic anomalies and peaks. At a global level, on the cloud centralized component, the local models can be merged in order to derive more high-level and refined statistics and prediction services that concern the behavior of users on a long-term basis, inferring seasonal (eg, daily or monthly) patterns. In this paper, we particularly focus on the local computation performed on the distributed nodes, with the objective of analyzing the performance in a variety of scenarios. The parallelization of the computation is achieved by partitioning the avenue into N regions and by assigning each region to a computing entity or "node." In this context, since the computing nodes are distributed along the avenue, it is necessary to resort to lightweight devices, cheap and with low power consumption. Single board computers can be profitably exploited because they are compact devices that

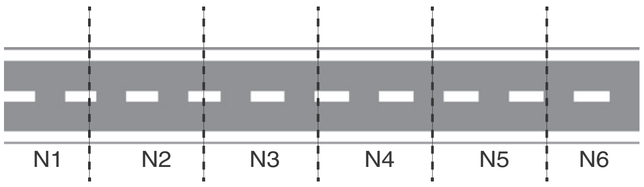

FIGURE 1 Smart avenue scenario

incorporate processor, memory, and I/O hardware on a single board. However, despite the reduced power consumption and the compact size, these mini-computers have a computing power comparable to low-end PCs currently available on the market. The architectures of these devices support general-purpose operating systems such as Android, Windows CE, and many of the major Linux distributions, ensuring all of their software capabilities. The reference device for the considered scenario is the Raspberry PI, consisting of a RM11@700Mhz processor, a RAM 256 MB, USB and network interfaces, with 5W consumption.

Each node has detailed information about the behavior of the users included in the region and receives summarized information about the users located in a number of neighbor regions, ie, information about the number and type of mobile devices that will probably enter the local region. We define the *visibility radius*R_V as the number of regions, on each of the two sides, from which a computing node receives information. The computation is performed at every given interval of time, or time step, whose duration depends on the applications requirements. An essential requirement is that the duration of the time step is longer than the time needed by the nodes to perform the computation and transmit related data among them, so that the nodes are able to keep the pace and complete the computation in time, ie, before the beginning of the next step.

At the end of a time step, each computing node sends information about the local region to the computing nodes up to R_V regions away. Only when a node receives the information from all the neighbor nodes it can start predicting the internet traffic for the next time step. In the section devoted to performance results, we will examine the effect of the number of nodes and the visibility radius on the computation efficiency and on the speedup.

The scenario of interest, outlined in Figure 1, is built upon the following assumptions:

- The length of the avenue under consideration is L, which, unless otherwise stated, is set to 10 km in this work. The width of the avenue is a constant; therefore, all the quantities that are assumed to be proportional to the area covered by a section of the avenue are also proportional to the length of the section;
- To simplify the scalability analysis, all the N computing nodes are assumed to have the same computation power;
- The mobile devices belong to two classes: those held by pedestrians and those held by vehicles. They move along the two directions with equal probabilities, and their average speed is 50 km/h for vehicles and 5 km/h for pedestrians. Clearly, this is a very simple mobility model. It is possible to use much more complex models, such as those defined in Singh et al[16] and Göndör et al,[17] but we use a simple model for two reasons: (1) It is sufficient to understand the basic behavior of the system; (2) the analysis is not influenced and biased by additional assumptions that are often related to a specific domain or city;
- The time that would be needed by a single node to perform the computation at a single step for the entire avenue is T_{serial}, assumed to be equal to 10 minutes in the case that L=10 km;
- The computation load of the nodes follows the distribution of users among the corresponding regions. Specifically, the average "computation time," ie, the average time needed to perform the computation on a single node, denoted as T_{node}, is proportional to the number of users located in the corresponding region;
- The time needed to communicate (transmit and receive) data with the neighborhood nodes is negligible with respect to the computation time. This assumption is reasonable in the case that only summarized and aggregated data is communicated, for example, the number and type of mobile devices and the estimation about the global data that will be transmitted by such devices.

In our experiments, we assume both a uniform and nonuniform distribution of mobile users along the avenue. In the first case, the average time needed to perform the computation on a single node, T_{node}, is proportional to the length of the corresponding avenue portion, ie, $T_{node} = T_{serial}/N$. In the case of nonuniform distribution of users, if $U(i)$ is the fraction of users located in the region i, the average computation time at node i, $T_{node}(i)$, is $T_{node}(i) = T_{serial} \cdot U(i)$.

To analyze the case on nonuniform distribution, we refer to Tirachini and Hensher,[18] a study on the use of public transportation in the city of Sydney, Australia, where it is found that between 30% and 50% of the users concentrate on a portion covering from 10% to 20% of the examined route. Accordingly, we separate the avenue into two portions: a high density 10% portion that contains a fraction of users, denoted as P_u, ranging from 0.1 to 0.6, and a low density portion that contains the rest of the users. Also, we considered two different nonuniform scenarios, referred to as "static" and "dynamic." In the static scenario, the high density portion of the avenue is always the same, eg, because this is the part of the avenue that contains most shops or services. In the dynamic scenario, the high density portion changes with time, due to the simultaneous movement of a large number of mobile users, eg, when offices open/close, or at the beginning/end of a sports event.

The actual computation time at a node can fluctuate around the average value in according to a random distribution. The type of random distribution is very difficult to predict in advance, as it depends on the specific scenario, for example, on the number of devices, on the behavior of users,

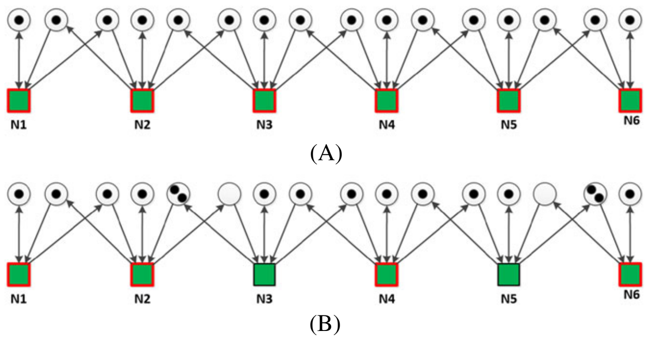

FIGURE 2 Petri net representing the execution of tasks at six parallel nodes, with R_V=1. In A, all the nodes are ready to execute. After execution at nodes N3, N4, and N5, the state of the Petri net is depicted in B: now N4 is ready to execute the next step, while N3 and N5 must wait for the execution at nodes N2 and N6, respectively

and on the number and type of other applications that are concurrently executed on the computing nodes. To generalize our analysis with respect to this aspect, we consider two random distributions: a gamma distribution with shape parameter equal to 2 and a lognormal distribution with shape parameter equal to 1. The gamma distribution is representative of distributions with relative low variability, and its probability density function has a similar shape as the normal distribution, except that negative values are not allowed. The lognormal distribution is representative of heavy-tailed distributions, with higher variability.

3 | PETRI NET MODEL FOR THE COMPUTATION

The parallel computation process defined by the described smart avenue application, can be represented by the Petri net model depicted in Figure 2, in a sample scenario with six parallel nodes and the visibility radius R_V set to 1. The Petri net formalism has been chosen because it allows to analyze issues related to the parallel and distributed nature of fog computing. Indeed, since the publication of the Carl Adam Petri's doctoral dissertation,[19] Petri nets are considered one of the first choices for the modeling and analysis of concurrent and distributed systems since they are particularly suitable to capture aspects such as synchronization, mutual exclusion and nondeterminism. In particular, a widely used variant of Petri nets, the Stochastic Time Petri nets,[20] is adopted in this work in order to model the variability of the computation time of complex smart city applications by means of purposely chosen random distributions.

As described in the previous section, we want to model a situation in which a node can proceed to the next step only after receiving some information from the neighbor nodes.

In the figure, six Petri net *transitions*, labeled as N1 to N6, are associated with the parallel nodes, and the *firing* of a transition corresponds to the execution of the computation at the corresponding node. Every transition is connected by inbound arcs to three input *places*, and in accordance to Petri net rules,[21] the transition is *enabled*, and the computation can start, if all the input places hold at least one *token*. When a transition *fires* (ie, the computation is performed at the current step), one token is *consumed* at each input place, and one token is *produced* on each of the output places, ie, the places connected to the three outbound arcs leaving the transition. One of these output place coincides with the input place of the same transition. The other two output places are input places of the two neighbor nodes: the production of a token on these two places models the delivery of the results to the neighbor nodes and the permission to such nodes to execute their computation at the next time step.[*]

Figure 2A represents the state of the system in a situation where all the Petri net transitions are enabled, ie, all the nodes are ready to execute the computation at the current step. The ability to perform the computation is represented by the presence of a red border on the square representing the transition. Figure 2B represents the situation after the execution of tasks at nodes N3, N4, and N5. N4 is now enabled to execute the next task, because it has performed the previous task and has received permissions by its neighbor nodes N3 and N5. In the Petri net model, this corresponds to the presence of three new tokens at the input places of N4, which means that the synchronization barrier which precedes the next computation at node N4 has been successfully passed. Conversely, nodes N3 and N5 are not yet enabled because they are still waiting for the completion of tasks at nodes N2 and N6, respectively.

Analogously, the scenario with R_V=2 is modeled by putting five input places at every transition and five outbound arcs that connect every transition to itself and to four neighbor nodes, two on the left and two on the right. The case of all-to-all synchronization among the nodes, where each node needs to receive the results from all the other nodes, is modeled with each transition preceded by *N* input places, and *N* outbound arcs connected to all the *N* nodes.

The Petri net model highlights the advantage of relaxing the synchronization requirements with respect to the classical parallel computation model, in which the synchronization involves all the nodes. In the case of "local synchronization," ie, when a node needs to synchronize with a limited number of neighbor nodes, different nodes are allowed to execute different time steps. For example, with R_V set to 1, each node can be one step

[*]The two transitions that correspond to the two extreme regions of the avenue are modeled differently, as depicted in the figure, and only two outbound arcs depart from those transitions.

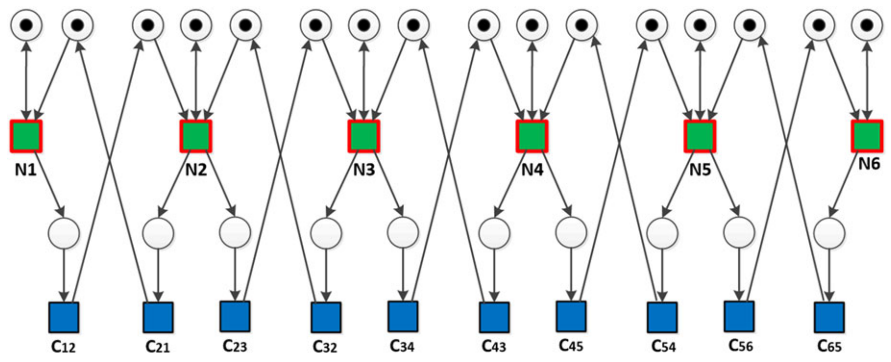

FIGURE 3 Petri net representing the execution of tasks at six parallel nodes, with R_V=1. Communication times are included and are modeled by transitions colored in blue

ahead than its direct neighbors, and the gap between the time steps executed by the two nodes located at the two ends of the avenue can be as large as *N*. This is a notable advantage in the case that the computation time varies from node to node and from step to step, as in the smart avenue case. The advantage resides in the fact that a longer execution time at one node does not slow down the execution at all the other nodes, but only at the neighbor nodes. As an example, if the nodes located at one end of the avenue are slower for a period of time (eg, due to the presence of a larger number of vehicles), the nodes located at the other end can proceed and execute some additional time steps. Successively, the nodes that are some steps behind can become faster and reach the other nodes, and so on. In the case of nonuniform distribution of users along the avenue, the advantage of local synchronization must be evaluated with care, as it can depend on the degree of imbalance and on the mobility of users. This will be discussed in the next section, devoted to the analysis of performance results.

The assumption that the communication time is very low with respect to the computation time, and thus can be ignored, is based on the fact that each node sends only summary data to the adjacent nodes. The validity of this assumption has been confirmed in several studies (ie, other works[22,23]), which experimentally show that the time needed to send local or summarized information is negligible with respect to the computation time in a large subset of distributed applications. The Petri net formalism, however, can also be used to model a more general scenario in which communication times should be explicitly taken into account. A Petri net that models this case, again for the case that R_v=1, is depicted in Figure 3. In the figure, "communication" transitions are colored in blue and each computation transition is connected to two communication transitions that explicitly model the time needed to transmit the data from the local node to the two neighbor nodes.

The Petri net model allows us to formally define the expression of the wallclock time experienced at a generic node $i \in \{1, 2, \dots, N\}$ after completing the execution at step *k*. The wallclock time, denoted as $T_i(k)$, is determined by the recursive expression:

$$T_i(k+1) = max(T_{i-1}(k) + C_{i-1,i}(k) + T_i(k) + T_{i+1}(k) + C_{i+1,i}(k)) + Tnode_i(k+1), \tag{1}$$

where $Tnode_i(s)$ is the time taken by node *i* to perform the computation at step *s* and $C_{m,n}(s)$ is the time needed to send data from node *m* to node *n* after the execution of step *s*. It is assumed that all the nodes begin the execution at the same time, ie, $T_i(0) = 0$ for each node *i*. To let the expression be consistent for the nodes at the two extremes, $T_0(s)$ and $T_{N+1}(s)$ are set to 0 for each step *s*. We used this expression to simulate the Petri net with Matlab, as discussed at the beginning of Section 4. The case in which communication times are negligible is managed by setting all the times $C_{m,n}(s)$ to 0.

4 | PERFORMANCE RESULTS

The aim of this section is to analyze the performance of the sample parallel application—the prediction of Internet traffic—in the described smart avenue scenario, when varying the two main design parameters: the overall number of nodes and the visibility radius. The computational performance is evaluated by measuring the *average step time*, defined as the wallclock time divided by the number of executed steps, and the related speedup with respect to the sequential scenario, in which computation is performed on a single node. In Section 4.1, we examine performances in the case of uniform distribution of users. Then, in Section 4.2, we discuss the trade-off between efficiency and accuracy of computation, again for the case of uniform distribution. Subsequently, in Sections 4.3 and 4.4, we consider the more general case in which a portion of the avenue contains a higher density of users with respect to the rest of the avenue. This is done in two specific scenarios, as anticipated in Section 2: a "static" scenario, in which the high density portion of the avenue is always the same, and a "dynamic" scenario, in which the high-density portion changes with time. In Section 4.5, we provide a final comment on the results.

The results have been obtained in two ways: (1) we used the well-known software Yasper[24] to reproduce the Petri net described in Section 3, and its "automatic simulation" tool to analyze the performance; (2) we wrote an ad hoc simulator in Matlab, which reproduces the same computation modeled by the Petri net, and computed the wallclock time using Expression 1. After verifying that the results are very close to each other, with the correlation factor always larger than 0.99, we chose to use Matlab, as its execution is faster, so we report the Matlab results in this section. As

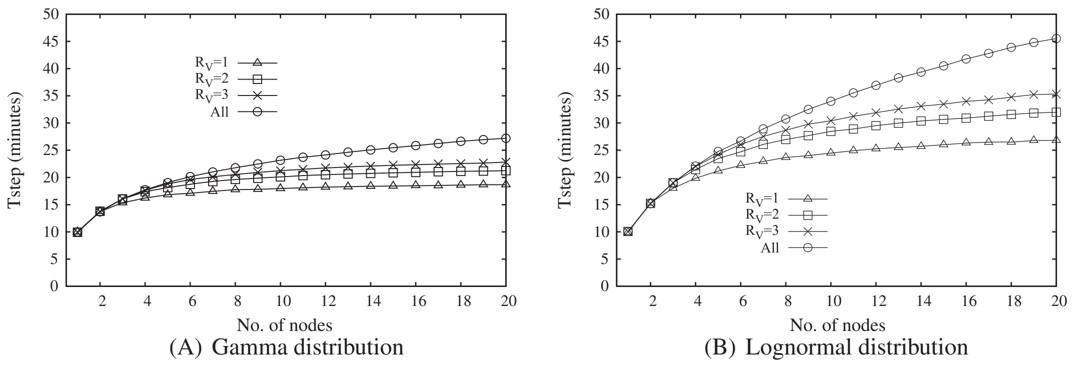

FIGURE 4 Values of T_{step} vs the number of regions, in the case that all the regions have the same length and the average computation time for a single region is 10 minutes

anticipated at the end of Section 2, all the results have been obtained by assuming two different random distributions for the computation time T_{node}, ie, the time needed to execute a step at a single node: a gamma distribution and a lognormal distribution.

4.1 | Performance with uniform distribution of users

In the first set of simulations, we examined the case in which the mobile users are uniformly distributed over the avenue and, as a consequence, the average time to execute a step is the same at all the computing nodes. To better understand the effect of the number of nodes and the visibility radius, we first considered a scenario in which the number of regions is varied and the length of each region is constant. This is a so-called *weak scalability* analysis, in which performances are investigated when varying the problem size, the length of the avenue in our case. We also tested three different values of the visibility radius R_V, from 1 to 3, and considered the case of all–to–all synchronization as a reference, ie, the visibility radius extends over the entire avenue. For this scenario, the average computation time experienced at a single node, T_{node} is fixed and set to 10 minutes.

We simulated the computation for a time equal to 30 days and obtained the average step time, T_{step}, by dividing the 30-day time interval by the number of completed steps. The results are shown in Figure 4, where the all-to–all synchronization case is indicated with the label "All." Figures 4A and 4B were obtained when using, respectively, the gamma distribution and the lognormal distribution to model the variability of the computation time at a single node.

The results show that, with a given value of N, the computation is faster (ie, the average time to execute a step on all the nodes, T_{step}, is shorter) when the synchronization is limited to a small number of neighbor nodes. It is also noticed that the value of T_{step} increases with the number of nodes N. In the "All" case, this happens because the time to execute one step on all the nodes is equal to the execution time of the slowest node, which corresponds to the maximum of N identical distributed random variables (in our case, gamma and lognormal random variables). This maximum value clearly increases with N.[†] When the synchronization involves a limited number of neighbor nodes, the effect of N on the value of T_{step} is more indirect, and can be illustrated as follows. If we take the case that R_V is set to one, a node n can execute a step t only when the nodes $n-1$ and $n+1$ have executed the step $t-1$. In turn, the node $n+1$ can execute the step $t-1$ when the node $n+2$ has executed the step $t-2$, etc. Therefore, the computation on one node is influenced, and possibly delayed, by the computation at all the other nodes, but this influence is experienced at different times, depending on the distance. Then, even if the induced delay tends to increase with the number of nodes, leading to larger values of T_{step}, the rate of the increase is much smaller than in the case of all–to–all synchronization, as clearly noticeable in Figure 4.

After this first set of experiments, we performed a *strong scalability* evaluation,[‡] ie, we focused on the case that the avenue, whose length L is fixed and set to 10 km, is partitioned into a number of regions N. As described in Section 2, the average computation time at a single node, T_{node}, is assumed to be proportional to the length of the region, $l=L/N$, and it is equal to T_{serial}/N, with T_{serial} set to 10 minutes. Figure 5 reports the values of the average step time T_{step}, when using the two random distributions for T_{node}. When N increases, the value of T_{step} decreases because the computation is partitioned among a larger number of nodes.

Figure 6, reporting the speedup—ie, the ratio between T_{serial} and T_{step}—is more useful to analyze the scalability. Actually, we found that the speedup value greatly depends on the type of random distribution of the computation time, specifically on its variability. If the distribution has a larger variability, the overhead time needed for the synchronization increases, and the speedup is lower, because there is a larger probability that one of the nodes takes significantly more time than the average, which forces the other nodes to wait at the synchronization barrier. On the other hand, the relative improvement that can be obtained when restricting the synchronization to a few neighbor nodes, is higher when the variability of the distribution is larger. As an example, when N is set to 20, in the case of gamma distribution, the speedup is equal to about 7.3 with all-to-all synchronization,

[†] Actually, it is known from the extreme value theory that the expected maximum of n identically distributed random variables is unbounded and grows as $\sqrt{2 \cdot ln(n)}$, for large n, for a set of widely adopted random distributions, including the exponential, the gamma, the normal, and the lognormal distributions.[25]

[‡] Strong scaling investigates, for a fixed problem size, how the time to solution varies with the number of processors. Weak scaling, on the other hand, studies how the time to solution varies with processor count with a fixed problem size per processor. These definitions can be found at www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

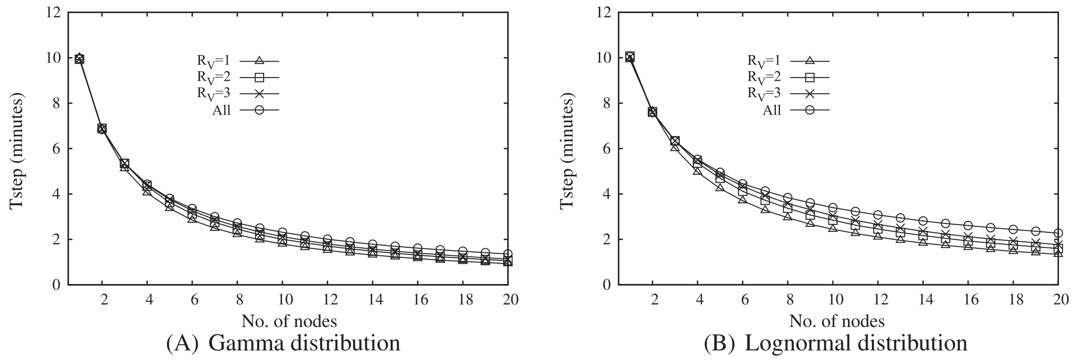

FIGURE 5 Values of T_{step} in the case of an avenue partitioned among a variable number of regions

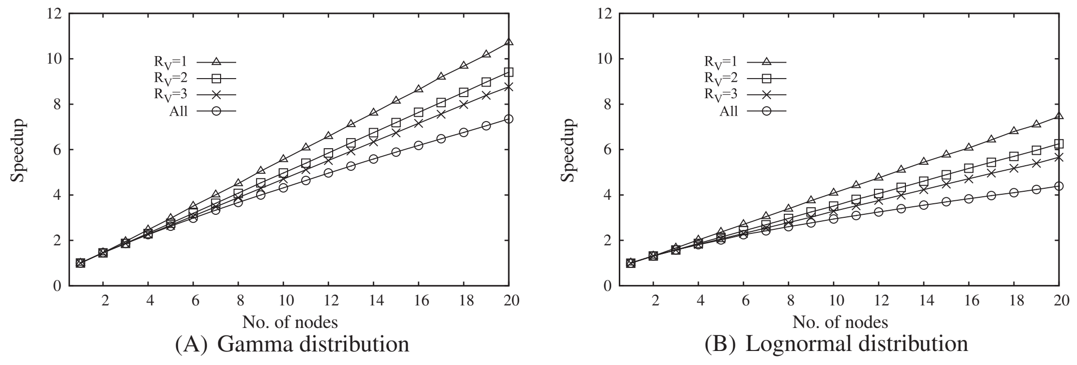

FIGURE 6 Values of the speedup in the case of an avenue partitioned among a variable number of regions

and to about 10.7 with R_V equal to 1, with a 46.5% improvement. In the case of lognormal improvements, the analogous speedup values are 4.41 and 7.40, with a 67.8% improvement.

4.2 | Speedup-accuracy trade-off with uniform distribution of users

When setting the number of nodes N and the visibility radius R_V, a trade-off emerges between minimizing the computation time and maximizing the accuracy of the computation. Indeed, parallelizing the computation on a larger number of nodes reduces the time T_{node} and therefore the time to complete the parallel computation related to a single step. However, a larger number of nodes corresponds to smaller regions: It means that the input data used by the computation is related to a smaller portion of the avenue (if the value of R_V is kept constant) and a smaller fraction of involved mobile devices, which can lead to a reduced accuracy of the results.

The second important trade-off concerns the value of R_V. On the one hand, a higher value of R_V is expected to slow down the computation, as also discussed in Section 4.1, due to the stronger impact of the involved *synchronization barrier*. Indeed, before executing the computation at step s, a node n must wait until $2 \times R_V$ neighbor nodes terminate their computation at step $s - 1$ and send to n the related computation results. The time needed for the synchronization is expected to increase with the number of involved nodes, $2 \times R_V$. On the other hand, a larger value of R_V (if the value of N is kept constant) allows the accuracy of the computation to be increased, because the computation can be based on information about a larger portion of the avenue.

To predict the Internet traffic that will be originated in a region during a time interval, it is necessary to consider not only the mobile devices already located in the region but also those that will arrive or transit during the time interval of interest. In a time interval T, a mobile device traveling with average speed v can travel a distance $s = v \times T$, and the number of regions of length l=L/N that can be traversed during T is $\lceil s/l \rceil = \lceil \frac{v \times T \times N}{L} \rceil$. Therefore, mobile devices can arrive, considering the two possible directions, from a number of regions equal to $N_R = min(N, 2 \times \lceil \frac{v \times T \times N}{L} \rceil)$. On the other hand, the number of "visible" regions, ie, the number of the neighbor regions that transmit data to the local region, is equal to $2 \times R_V$. We then define the *coverage ratioC*, or simply *coverage*, as the ratio between the number of visible regions and the number of regions from which mobile devices can arrive:

$$C = \frac{2 \times R_V}{N_R}. \tag{2}$$

This ratio is used as a measure of the accuracy of the prediction. Indeed, the coverage ratio equal to 100% means that the computation is able to consider the data related to all the mobile devices that can arrive or pass through the local region. When the coverage is lower than 100%, however,

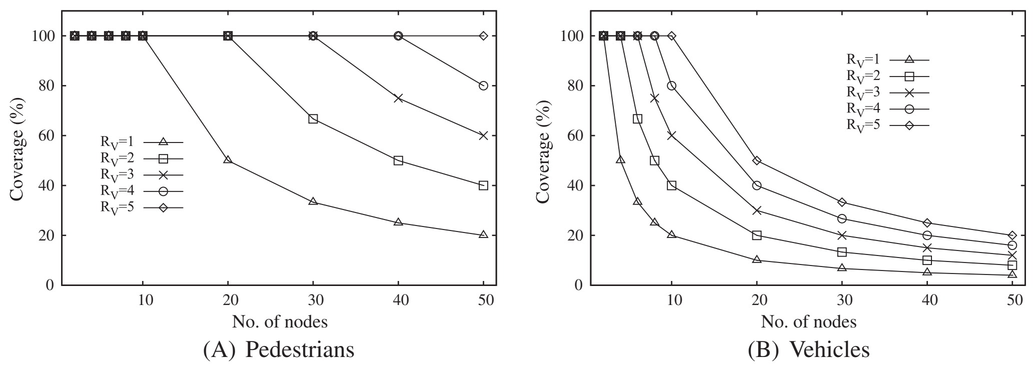

FIGURE 7 Coverage ratio for mobile devices held by A, pedestrians and by B, vehicles

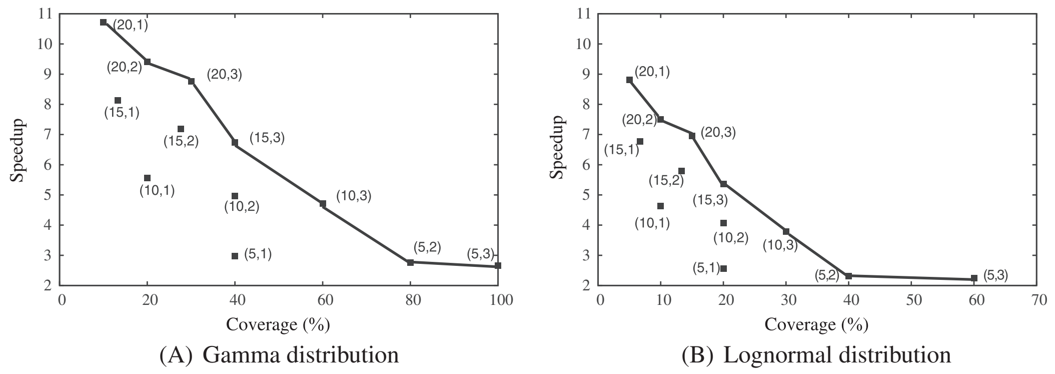

FIGURE 8 Values of coverage and speedup for different values of the couple (N, R_V). The Pareto frontier is shown

the computation does not receive information from some neighbor regions from which mobile devices can actually arrive, and the computation can be less accurate.

Of course, the coverage is always equal to 100% in the case of all-to-all synchronization, since each node receives information from all the other regions. In all the other cases, the value of C depends on the speed of mobile devices, the number of nodes N and the visibility radius R_V. Figure 7 shows the values of the coverage ratio computed for the devices held, respectively, by pedestrians traveling at 5 km/h and by vehicles traveling at 50 km/h, in the case that the length L of the avenue is 10 km and the time interval T is set to 10 minutes. Of course, with the same values of N and R_V, the coverage is lower for vehicles than for pedestrians, as vehicles can reach farther regions in the same amount of time, and the value of N_R, in the denominator of Expression 2, is higher. In addition, it clearly appears that the coverage decreases with larger values of N and with smaller values of R_V. Figure 7 can be used by administrators to set the value of parameters needed to achieve a desired goal with given constraints. For example, if N is set to 10, the two figures show that the value of R_V must be set to a value equal or larger than 3 if the desired coverage is at least 50% for both vehicles and pedestrians.

As speedup and coverage are heterogenous objectives, they cannot be easily combined in a single optimization function. However, the analysis of Pareto frontiers can help to tune the values of the parameters, in our case N and R_V. Figure 8 reports the values of speedup and coverage, measured for vehicles, obtained with different values of the couple (N, R_V) and shows the Pareto frontiers. The two figures are obtained by using the two considered random distributions for the computation time T_{node}. Values of N and R_V that are not positioned on the frontier are not acceptable, because other choices of the parameter values allow both the objectives to be improved. Values that are positioned on the frontier, however, can be considered by the administrator and can be chosen depending on the relative importance of the two objectives.

4.3 | Performance with static nonuniform distribution of users

So far, we considered the case in which the mobile users are uniformly distributed. To analyze a more general scenario, as mentioned in Section 2, we separate the avenue into two portions: a high density 10% portion that contains a fraction of users, denoted as P_u, ranging from 0.1 to 0.6, and a low density portion that contains the rest of the users. The distribution of users is assumed to be uniform within each of the two portions. In this section, we consider the case of a static scenario, where the density of users does not change with time. The dynamic scenario will be considered in Section 4.4. It is recalled here that the average computation time at a node i is directly proportional to the fraction of users located in the local region, $U(i)$, ie, $T_{node}(i) = T_{serial} \cdot U(i)$.

We first examine the scenario with P_u equal to 0.4. Figures 9 and 10 report the values of the average step time and of the speedup, respectively, vs the number of nodes among which the smart avenue is partitioned. As usual, the results are obtained when using the two considered random

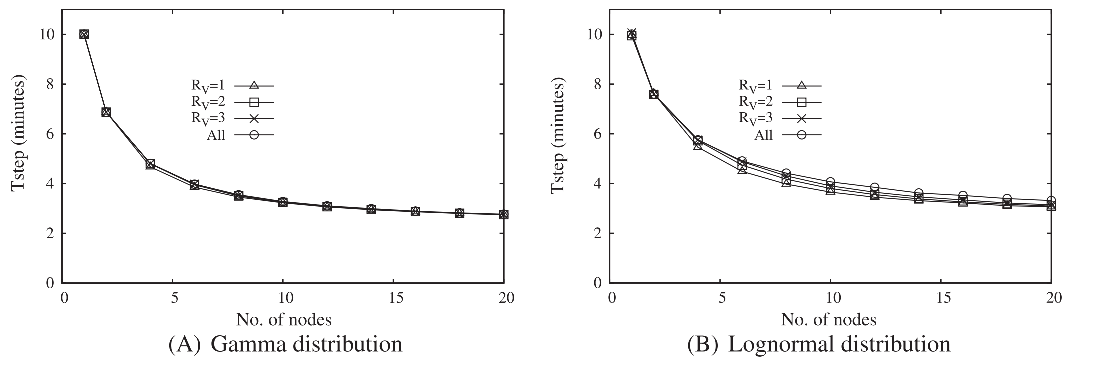

FIGURE 9 Values of T_{step} in the case of an avenue partitioned among a variable number of regions, with P_u=0.4

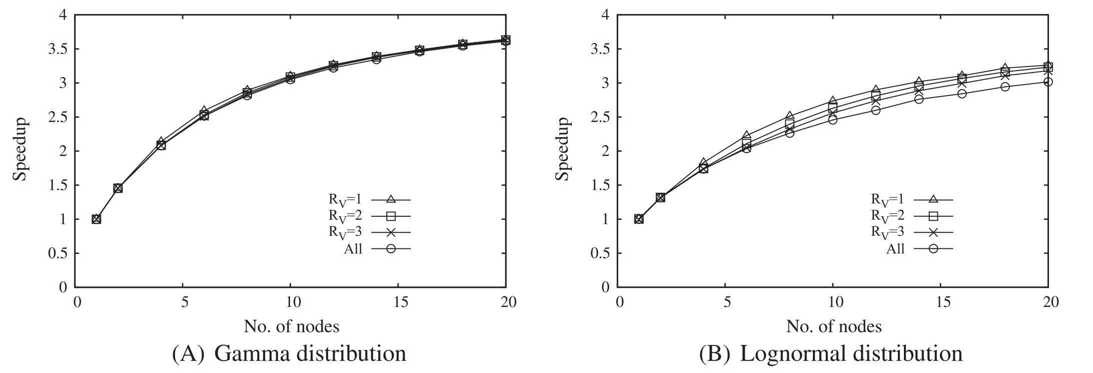

FIGURE 10 Values of the speedup in the case of an avenue partitioned among a variable number of regions, with P_u=0.4

distributions for T_{node}. Several considerations can be done, when comparing these results to those obtained with uniform user distribution, reported in Figures 5 and 6. They can be summarized as follows:

1. With nonuniform user distribution, the average step time is longer, and the speedup is lower. The reason is that it is extremely likely that the regions with higher density of users act as a bottleneck, ie, they take more time to execute the computation and the other nodes are obliged to wait at the synchronization barrier;
2. The speedup is not only lower but increases at a much lower rate with respect to the number of nodes;
3. There is some benefit that can be obtained with local synchronization, with respect to all-to-all synchronization, but this benefit is much lower than in the case of uniform user distribution. In other words, the presence of some nodes that are always much slower than the others reduces the advantage of local synchronization;
4. In any case, the advantage of local synchronization is still remarkable in the case of lognormal distribution of the computation time, while it is almost negligible with gamma distribution.

Figure 11 shows the values of T_{step}, with a number of nodes set to 20, gamma distribution, and with different values of P_u. On the left, we report the absolute values of T_{step}; on the right, we show the values normalized to those obtained with all-to-all synchronization, in order to better emphasize the advantage that can be obtained with local synchronization. Analogous results are reported in Figure 12, for the case of lognormal distribution. In the figures on the left, reporting absolute values, it is confirmed that the efficiency of computation decreases when more users are concentrated in the high density section. It also appears that the advantage achieved with local synchronization decreases when increasing P_u. This second effect is much more evident in the figures on the right, and it is larger with lognormal distribution.

From these results, it can be derived that the scenario with nonuniform distribution of users must be tackled with care. The most natural approach is to adopt a different distribution of nodes on the avenue and assign more nodes to the high density regions, so as to balance the load and return to a scenario in which the average computation times are comparable on the different nodes. It should be noted, however, that this approach can only be efficient in a steady situation, in which some portions of the avenues are always more populated than the others. In a more dynamic scenario, such a strategy would require to dynamically modify the number and position of the nodes along the avenue, which of course is a much more complex and expensive solution.

Fortunately, the results of the next section show that in a dynamic scenario, where the regions that sustain the highest load change with time, it is not necessary to modify the allocation of computational nodes to ensure a good degree of efficiency.

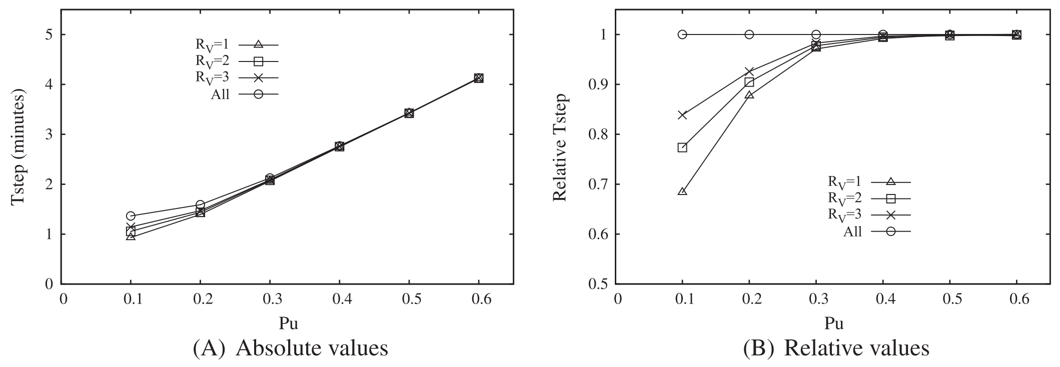

FIGURE 11 Values of T_{step} vs the value of the fraction P_u, with $N = 20$, gamma distribution. A, Absolute values; B, values normalized w.r.t. the case of all-to-all synchronization

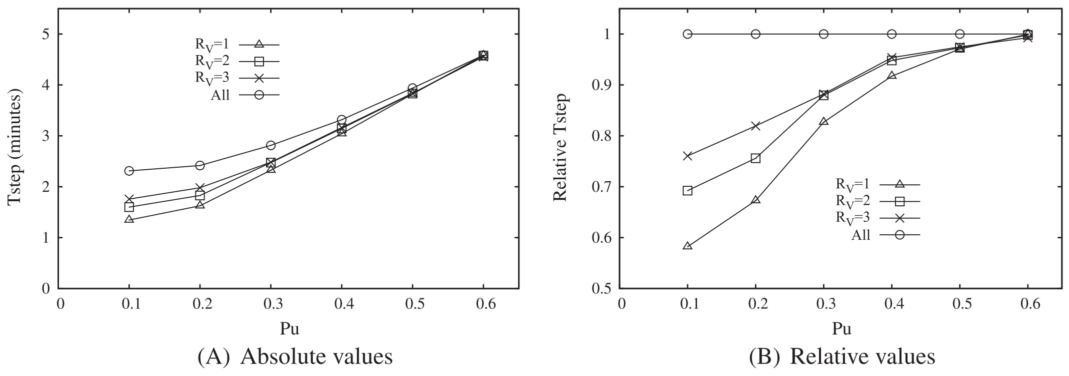

FIGURE 12 Values of T_{step} vs the value of the fraction P_u, with $N = 20$, lognormal distribution. A, Absolute values; B, values normalized w.r.t. the case of all-to-all synchronization

4.4 | Performance with dynamic nonuniform distribution of users

In the previous section, we have assumed that the distribution of users is steady, even when it is not uniform. In general, however, this assumption does not hold, for example, when many users simultaneously move along the avenue. This can happen before or after an event, when offices open/close, or simply due to the normal variations of user distributions during the day. To analyze this scenario, we performed a set of experiments in which the high density region, covering 10% of the avenue, moves along the avenue itself. Specifically, we start from a situation where the high density region, also referred to as the "peak" in the following, is located at the extreme left of the avenue, and then it moves with a constant speed to the right. To give a statistical reliability to the results, the peak inverts the movement when it arrives at any of the two extremes, for all the duration of the experiment. The speed of the peak, denoted as v, is defined as the fraction of the avenue traversed by the peak in an hour.

Figure 13 reports the values of T_{step} obtained when using N nodes, in the case that the high density region contains 40% of mobile users, ie, $P_u = 0.4$, and the computation is distributed among 20 nodes. The speed of the peak, v, is varied between 0 and 0.9. It is interesting to notice that the velocity has nearly no effect in the case of all-to-all synchronization. The reason is that, though the nodes with the highest load are different at different steps, the longer time they need to complete the execution obliges all the other nodes to wait for them at synchronization barriers. Conversely, with local synchronization the most loaded nodes only slow down the neighbor nodes, while the farther nodes can proceed faster. Since the peak moves along the avenue, this effect involves different nodes at different times, which allows the overall execution to be more efficient: The nodes that are slowed down at a certain step can be faster at successive steps, and vice versa. As an example, with $v = 0.5$, and lognormal distribution, the value of T_{step} is about 3.65 minutes with all-to-all-synchronization and reduces to about 1.6 minutes with $R_v = 1$, which corresponds to a percentage improvement of about 56%.

In Figure 14, we report the values of speedup obtained with $P_u = 0.4$, when the velocity of the peak is $v = 0.5$. From a comparison with Figure 6 and Figure 10, we see that the values of speedup are still worse than those obtained with uniform distribution of users, but they are much better than the values experienced with nonuniform but static distribution. Also, we notice that local synchronization offers a notable advantage with respect to all-to-al synchronization. In this case, however, the advantage is specifically observed with $R_v = 1$, while it is much lower with $R_v = 2$ and $R_v = 3$. The reason is that with a larger value of the visibility radius, it is more likely that the peak of mobile users involves and slows down the same nodes at more consecutive steps, causing an overall delay of the execution.

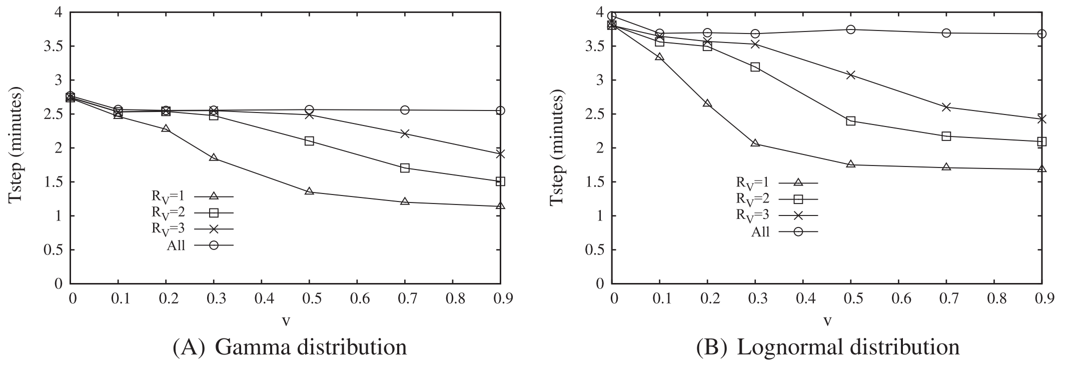

FIGURE 13 Values of T_{step} vs the value of the peak velocity v, with $P_u = 0.4$, $N = 20$

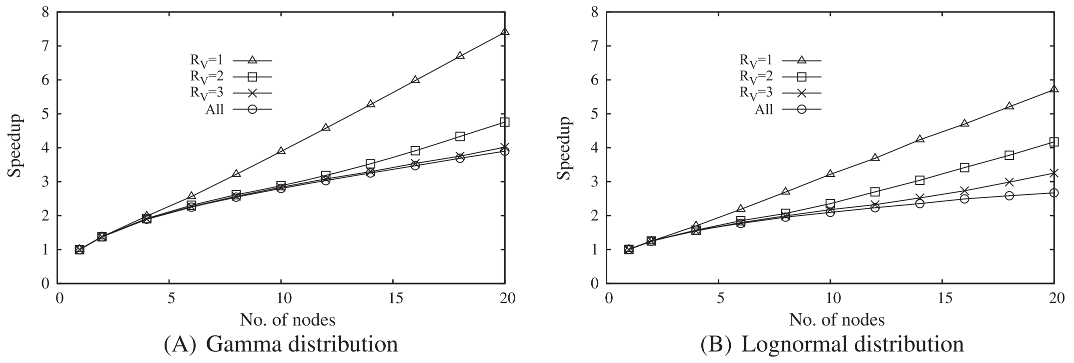

FIGURE 14 Speedup with $P_u = 0.4$ and $v = 0.5$

4.5 | Summary of results

Some interesting conclusions can be derived from the achieved results:

- We analyzed the local synchronization approach, based on the fact that a node is required to receive and send information only to a set of neighbor nodes, corresponding to adjacent sections of the avenue. We found that this approach offers a notable advantage with respect to all-to-all synchronization, in terms of computation efficiency and speedup, especially in the case of uniform distribution of users;
- Conversely, in the case of nonuniform and static distribution of users, we found that the advantage of local synchronization is minimal, and decreases as the imbalance of users distribution increases. In this scenario, the best solution is to install more computing nodes in the high density regions, in order to re-establish the load balance;
- With nonuniform and dynamic distribution of users, however, the efficiency and speedup obtained with local synchronization increases again. In this scenario, it is better to allocate the computing nodes as in the case of uniform distribution. Another viable solution, if there are many available computing nodes, can be to switch the nodes on or off depending on the current distribution of users, for example, by activating more nodes when there are more users. This solution, however, needs to be evaluated carefully;
- We tested two different random distributions of the computational node experienced at a single node, having the same average but different variability: the gamma and the lognormal distributions. We found that performances are generally poorer when the variability is higher (ie, with the lognormal distribution) but also that the relative advantage of local synchronization increases;
- For the case of uniform user distribution, we performed a preliminary analysis of the trade-off between efficiency and accuracy, when varying the overall number of nodes and the number of nodes involved in the local synchronization. In future work, this analysis can be refined by using more detailed mobility models and can be extended to the case of nonuniform distribution.

The results concern a relatively stable scenario, in which the long-term behavior of users, in terms of mobility and internet traffic, is consolidated. If such behavior changes significantly, for example, due to modifications in the urban environment, the deployment of the distributed computing nodes could become unsuitable. In this case, two solutions can be adopted: the first is to redraw the deployment, which can lead to significant financial efforts; the second is to resort to the over provisioning of distributed nodes and activate the nodes dynamically depending on the current distribution and behavior of users.

5 | RELATED WORK

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 areas including wireless and sensor networks, mobile and distributed computing, embedded systems, agent technologies, autonomic communication, and cloud computing. The variety of involved application domains is also wide [26]: transportation and logistics, smart electrical grids, big data and business analytics, social sciences, etc.

One of the most relevant issues in the IoT context is the appropriate repartition of computation and storage tasks among the so-called *smart devices*, distributed on the territory, and the more powerful centralized servers, for example, those of a cloud data center. This aspect is examined in Redondi et al,[9] where the authors evaluate the performance of a visual sensor network and consider the pros and cons of two alternative approaches for computation, ie, "compress-then-analyze" versus "analyze-then-compress." With the first option, images are acquired and compressed by local sensors, and then transmitted to remote computation nodes for analysis. On the other hand, the second strategy aims to reduce the bitstream flowing by resorting to a preliminary analysis performed on local nodes, in the case that these have enough power to be able to extract and encode complex high-level visual features autonomously. In Kavalionak et al,[10] the authors focus on a video surveillance application. The video elaboration is carried out in parallel by means of distributed devices that elaborate the camera video stream locally and, when local computational resources are not sufficient, send data to a more powerful centralized server.

The intelligent management of "smart cities" is one of the most important application scenarios of the Internet of Things paradigm. The challenge is to harness the collaborative power of ICT networks (networks of people, of knowledge, and of sensors) and use the resulting collective intelligence to implement better informed decision-making processes and empower citizens, through participation and interaction, to adopt more sustainable individual and collective behaviors and lifestyles.[4,27] The collection of mobile phone data and the discovery of mobility models is one of the most challenging issues in urban computing and can help to achieve several smart city objectives. For example, transportation analysis through mobile phone data can be applied for estimating road traffic volume and transport demands, to forecast public's future demand for taxis, to infer origins of tourists and visitors, to produce movement patterns to support city managers in transport planning, intelligent traffic management, route recommendations, etc.[28-30] Several smart social mobility services are also depicted in Sassi and Zambonelli,[31] which include smart parking services, itinerary matching and ride sharing, taxi sharing, multimodal rides, and chaperone services.

In this paper, we present and evaluate a model for the efficient parallel execution of a sample smart city application, ie, the analysis and prediction of the data and Internet traffic generated by a large number of mobile users.[12-14] The analysis of Internet traffic generated by mobile users is an important application scenario today,[16,17,32,33] as numerous vehicles possess powerful sensing, networking, communication, and data-processing capabilities and can exchange information with each other (vehicle to vehicle, V2V) or with the roadside infrastructure such as camera and street lights (vehicle to infrastructure, V2I) over various protocols, including HTTP, SMTP, TCP/IP, WAP, and Next Generation Telematics Protocol (NGTP).[15]

The rapid and correct prediction of Internet data traffic has several applicative impacts in smart city scenarios.[16,17,34,35] A first example concerns to the so-called *infotainment applications*, such as community services (ie, point-of-interest notifications, electronic and financial services, media downloading, and parking zone management), which can require the transmission of a huge amount of data among users and road units for an efficient provisioning of such services.[34] A second application, recently described in Altomare et al,[36] is the accurate prediction of Internet traffic, which is useful to anticipate possible bottlenecks in some portions of the avenue, and save energy and batteries consumption by dynamically redistributing the workload between fixed and mobile devices.[35] Finally, usage pattern prediction of requests can be also used to influence the admission/denial of service demands made by priority and nonpriority users, in order to match their respective Quality of Service agreements. Sometimes this is also supported by the use of machine learning algorithms for traffic behavior modeling of mobile users, which is becoming a challenging issue to improve service effectiveness and efficiency.[16,17]

The parallel analysis of the data and knowledge extracted from urban environments is a key element to enhance quality, improve performance and safety of urban services, and reduce costs and city resource consumptions. For instance, we can detect the root cause of urban air pollution by studying the correlation between air quality and the data concerning traffic flows and hot spots where a large density of mobile users are concentrated.[1,30] Another interesting application allows us to individuate underlying problems in a city's road network, and to better formulate city planning for the future,[1] through the analysis of city-wide human mobility data.

Some of the main technical challenges that must be tackled are the high mobility of vehicles, the wide range of relative speeds between nodes, and the real-time nature of applications.[33] The efficient execution of parallel applications can be essential to support such challenges and to ensure the timely provisioning of application results. For example, active road safety applications primarily provide information and assistance to drivers to avoid such collisions with other vehicles. This can be accomplished by rapidly sharing and processing information between vehicles and road side units. Information can represent vehicle position, intersection position, speed, and distance heading. Traffic efficiency and management applications, aiming at improving the traffic flow, coordination, and assistance, also require the timely delivery of updated local information, maps and in general, messages of relevance bounded in space and/or time, and can benefit from efficient execution on a parallel/distributed infrastructure.[2]

An important aspect to be considered is related to fault tolerance management, which is a critical issue in a large-scale distributed architecture like the one presented here. In fact, each node has sensing and processing capability, sometimes deployed in hostile and harsh conditions, eg, rain, snow, wind, and hence susceptible to periodic and unexpected errors. Generally, faults in a wireless sensor network are classified into two categories: (1) persistent faults and (2) transient faults.[37] Transient faults are temporary faults that occur in certain conditions such as network congestion and bad weather. Techniques for detecting transient faults are discussed in other works[38,39] and are generally based on the checkpoint-recovery strategy. Persistent faults persist until a fault recovery action is performed. In the majority of cases, faults are local in the network and only affect a few components of the network.[40]

The prediction of Internet traffic that we study in this work is highly sensitive to two main factors: (1) the distribution of people and (2) the mobility of vehicles along the avenue. For what concerns the first issue, this study analyzes both the case in which the mobile users are uniformly distributed and the case in which they are not. Specifically, we refer to Tirachini and Hensher,[18] a study on the use of public transportation in the city of Sydney, Australia, which found that between 30% and 50% of the users concentrate on a portion covering from 10% and 20% of the examined route. For what concerns the mobility of vehicles, in this paper, we use a simple model assuming that mobile users belong to two classes, pedestrians, and vehicles, and move with two different average speeds. This assumption aims to make results general and independent from the specific scenario. Whenever our model is applied to a specific real-world case—ie, a specific urban area, city, or district—we can exploit the mobility models discovered through some data-driven approach, as described in previous studies.[41,42] Some examples of more complex mobility models are defined in Singh et al[16] and Göndör et al.[17]

6 | CONCLUSION AND FUTURE WORK

This paper addresses the issue of efficiently managing the parallel and concurrent execution of smart city applications, where the computation is driven by space-aware information. We focused on the sample mono-dimensional scenario of a city avenue where the objective is to analyze the Internet traffic generated by vehicles and pedestrians. The strategy is to distribute the computational load among a number of nodes, where each node is assigned to a portion of the avenue and exchanges information with the nodes assigned to neighbor portions. However, the efficiency and the effectiveness of the computation strictly depend on the synchronization paradigms applied among the nodes. In this paper, we have considered two synchronization mechanisms: local synchronization (a limited number of parallel nodes need to synchronize before proceeding to the next computation step) and all-to-all synchronization (all nodes must synchronize among them). In particular, we evaluated how the efficiency and speedup obtained with the two synchronization paradigms are influenced by (1) different distributions of mobile users along the avenue and (2) different probability distributions of the computational times experienced at the single nodes. We showed that is possible to tune some system parameters—in particular, the overall number of computing nodes and the number of neighbor nodes among which the information is transmitted—to achieve the desired trade-off between the efficiency and the accuracy of computation. Specifically, when information is exchanged among a larger number of nodes, the overall computation time increases but the accuracy of computation is enhanced, and vice versa. Moreover, we presented a Petri net that we have exploited to model and analyze the smart avenue and can be easily adapted to different scenarios. Current work aims at extending the analysis to bi- and three-dimensional scenarios, and at investigating the applicability of our approach to other important applications, for example, smart energy applications in which several *prosumers*, ie, entities that can consume and produce electricity, cooperate among them, and the electricity price set on one prosumer depends on the consumption and production of energy at a number of neighbor prosumers.

ACKNOWLEDGMENTS

This work is partially funded by the MIUR project “DOMUS,” no. PON03PE_00050_2.

ORCID

Carlo Mastroianni http://orcid.org/0000-0001-6269-4931

REFERENCES

1. Zheng Y, Capra L, Wolfson O, Yang H. Urban computing: concepts, methodologies, and applications. *ACM Trans Intell Syst Technol*. 2014;5(3):1-55.
2. Nellore K, Hancke GP. A survey on urban traffic management system using wireless sensor networks. *Sensors*. 2016;16(2):157.
3. Blecic I, Cecchini A, Trunfio GA, Verigos E. Urban cellular automata with irregular space of proximities. *J Cell Automata*. 2014;9(2-3):241-256.
4. Khatoun R, Zeadally S. Smart cities: concepts, architectures, research opportunities. *Commun ACM*. 2016;59(8):46-57.
5. Cicirelli F, Forestiero A, Giordano A, Mastroianni C, Spezzano G. Parallel execution of space-aware applications in a cloud environment. In: 24th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2016), Heraklion, Crete, Greece; 2016:686-693.
6. Cicirelli F, Forestiero A, Giordano A, Mastroianni C. Transparent and efficient parallelization of swarm algorithms. *ACM Transactions on Autonomous and Adaptive Systems*. 2016;11(2)Article no. 14.
7. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the Internet of Things. In: Proceedings of the 1st ACM MCC Workshop on Mobile Cloud Computing, Helsinki, Finland; 2012:13-16.
8. Krishnan YN, Bhagwat CN, Utpat AP. Fog computing- network based cloud computing. In: 2nd IEEE International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India; 2015:250-251.
9. Redondi A, Baroffio L, Bianchi L, Cesana M, Tagliasacchi M. Compress-then-analyze versus analyze-then-compress: what is best in visual sensor networks?. *IEEE Trans Mobile Comput*. 2016;15(12):3000-3013.
10. Kavalionak H, Gennaro C, Amato G, Meghini C. Dice: A distributed protocol for camera-aided video surveillance. In: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on IEEE: Liverpool, UK; 2015:477-484.
11. Mastroianni C, Cesario E, Giordano A. Balancing speedup and accuracy in smart city parallel applications. In: Proc. of Euro-Par Workshops, Grenoble, France; 2016:224-235.
12. Harri J, Filali F, Bonnet C. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Commun Surv Tutorials*. 2009;11(4):19-41.
13. Lochert C, Mauve M, Fussler H, Hartenstein H. Geographic routing in city scenarios. *SIGMOBILE Mob Comput Commun Rev*. 2005;9(1):69-72.

14. Menouar H, Lenardi M, Filali F. Movement prediction-based routing (MOPR) concept for position-based routing in vehicular networks. In: 2007 IEEE 66th Vehicular Technology Conference, Baltimore, MD, USA; 2007:2101-2105.
15. Hank P, Müller S, Vermesan O, Van Den Keybus J. Automotive ethernet: in-vehicle networking and smart mobility. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE '13); 2013; San Jose, CA, USA:1735-1739.
16. Singh R, Srinivasan M, Murthy CSR. A learning based mobile user traffic characterization for efficient resource management in cellular networks. In: 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA; 2015:304-309.
17. Göndör S, Uzun A, Rohrmann T, Tan J, Henniges R. Predicting user mobility in mobile radio networks to proactively anticipate traffic hotspots. In: Proceedings of the 2013 International Conference on Mobile Wireless Middleware, Operating Systems, and Applications (Mobilware'13); 2013; Bologna, Italy:120-129.
18. Tirachini A, Hensher DA. Bus congestion, optimal infrastructure investment and the choice of a fare collection system in dedicated bus corridors. *Transp Res Part B: Methodological*. 2011;45(5):828-844.
19. Murata T. Petri nets: properties, analysis and applications. *Proc IEEE*. 1989;77(4):541-580.
20. Paolieri M, Horvath A, Vicario E. Probabilistic model checking of regenerative concurrent systems. *IEEE Trans Software Eng*. 2016;42(2):153-169.
21. Peterson JL. Petri nets. *ACM Comput Surv*. 1977;9(3):223-252.
22. Cesario E, Mastroianni C, Talia D. Distributed volunteer computing for solving ensemble learning problems. *Future Gener Comput Syst*. 2016;54:68-78.
23. Cesario E, Talia D. Distributed data mining patterns and services: an architecture and experiments. *Concurrency Comput: Pract Experience*. 2012;24(15):1751-1774.
24. van Hee K, Oanea O, Post R, Somers L, an der Werf JM. Yasper: A tool for workflow modeling and analysis. In: Proceedings of the Sixth International Conference on Application of Concurrency to System Design (ACSD '06) IEEE Computer Society; 2006; Washington, DC, USA:279-282.
25. De Haan L, Ferreira A. *Extreme Value Theory: An Introduction*: Springer Science & Business Media; 2007.
26. Lee I, Lee K. The Internet of Things (IoT): applications, investments, and challenges for enterprises. *Bus Horiz*. 2015;58(4):431-440.
27. Mitton N, Papavassiliou S, Puliafito A, Trivedi KS. Combining cloud and sensors in a smart city environment. *EURASIP J Wireless Commun Networking*. 2012;2012(1):1-10.
28. Lee J-G, Han J, Li X. A unifying framework of mining trajectory patterns of various temporal tightness. *IEEE Trans Knowledge Data Eng*. 2015;27(6):1478-1490.
29. Zheng Y. Trajectory data mining: an overview. *ACM Trans Intell Syst Technol*. 2015;6(3):1-41.
30. Cesario E, Comito C, Talia D. Towards a cloud-based framework for urban computing, the trajectory analysis case. In: In Proc. of the 2013 International Conference on Cloud and Green Computing (CGC'13), Auckland, New Zealand; 2013:16-23.
31. Sassi A, Zambonelli F. Coordination infrastructures for future smart social mobility services. *IEEE Intell Syst*. 2014;29(5):78-82.
32. Sommer C, Dressler F. *Vehicular Networking*: Cambridge University Press; 2014.
33. Karagiannis G, Altintas O, Ekici E, Heijenk G, Jarupan B, Lin K, Weil T. Vehicular networking: a survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Commun Surv Tutorials*. 2011;13(4): 584-616.
34. Li Z, Wu P, Song Y. A fast reroute algorithm for infotainment service in internet of vehicles. *Int J Distrib Syst Technol*. 2016;7(3):63-77.
35. Li R, Zhao Z, Zhou X, Palicot J, Zhang H. The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice. *IEEE Commun Mag*. 2014;52(6):234-240.
36. Altomare A, Cesario E, Talia D. Energy-aware migration of virtual machines driven by predictive data mining models. In: Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2015); 2015; Turku, Finland:549-553.
37. Muhammed T, Shaikh RA. An analysis of fault detection strategies in wireless sensor networks. *J Network Comput Appl*. 2017;78:267-287.
38. Sharma KP, Sharma TP. rdfd: reactive distributed fault detection in wireless sensor networks. *Wireless Networks*. 2017;23(4):1145-1160.
39. Mahapatro A, Khilar PM. Online fault diagnosis of wireless sensor networks. *Central Eur J Comput Sci*. 2014;4(1):30-44.
40. Kutten S, Peleg D. Fault-local distributed mending. *J Algorithms*. 1999;30(1):144-165.
41. Altomare A, Cesario E, Comito C, Marozzo F, Talia D. Trajectory pattern mining for urban computing in the cloud. *IEEE Trans Parall Distrib Syst*. 2016;28(2):586-599.
42. Monreale A, Pinelli F, Trasarti R, Giannotti F. Wherenext: a location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM SIGKDD, KDD '09, Paris, France; 2009: 637-646.

How to cite this article: Mastroianni C, Cesario E, Giordano A. Efficient and scalable execution of smart city parallel applications. *Concurrency Computat: Pract Exper*. 2018;30:e4258. https://doi.org/10.1002/cpe.4258