Balancing Speedup and Accuracy in Smart City parallel applications

Eugenio Cesario, Andrea Giordano, Carlo Mastroianni

ICAR-CNR, Rende (CS), Italy
Email: {cesario,giordano,mastroianni}@icar.cnr.it

Abstract. Smart city and Internet of Things applications can benefit from the use of distributed computing architectures, due to the large number and pronounced territorial dispersion of the involved users and devices. In this context, a natural method to parallelize the computation is to consider the territory as partitioned into regions, e.g., city neighborhoods, and associate a computing entity with each region. The application considered in this paper is the prediction of the amount of internet traffic generated within a given region, which requires to consider not only the devices located in the region but also the mobile devices that are expected to enter the local region in the future. When setting the number of neighbor regions included in the computation, it must be considered that this parameter has opposite effects on two important objectives: increasing the number of neighbors tends to improve the accuracy of the prediction but slows down the computation because more computing entities need to synchronize among each other. Similar considerations apply when setting the size and number of regions that partition the territory. This paper offers an insight onto these important tradeoff issues.

1 Introduction

In the last few years, increasing attention is devoted to the field of the so-called "Internet of Things" (IoT), an emerging paradigm built upon the research and development advances in a wide range of areas including wireless and sensor networks, mobile and distributed computing, embedded systems, agent technologies, autonomic communication, Cloud computing. The variety of involved application domains is also wide [10]: transportation and logistics, smart electrical grids, big data and business analytics, social sciences, etc. The intelligent management of "smart cities" is one of the most important application scenarios of the Internet of Things paradigm. Sustainable development of urban areas is a challenge of key importance and requires new, efficient, and user-friendly technologies and services[3]. The challenge is to harness the collaborative power of ICT networks (networks of people, of knowledge, 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[12]. High-quality can be obtained by cross-correlating data retrieved from a number of sensors and objects and by analyzing such data with sophisticated algorithms.

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: for example, environmental information extracted from sensors, data inherent to the neighborhoods and

residential units in a city, etc. It is then natural to manage such data through the use of computing entities distributed over the territory, in order to perform the computation as close as possible to data sources and to improve the performances by increasing the degree of parallelization[4]. Cloud computing provides an ideal back-end solution for handling the data produced by such a large number of heterogeneous devices. However, because of the inherent dispersion of data and computing entities, it can be unfeasible or inconvenient to bring computation to a single Cloud infrastructure, e.g., a big centralized data center. A better support to tackle mobility and geo-distribution of data, embrace location awareness and ensure low latency, can be provided by a variant of Cloud, referred to as *Fog Computing* [2][9], which is composed by a number of distributed Cloud facilities located close to data sources, i.e., a cloud close to the ground. The computation related to smart city applications can be partitioned and parallelized by assigning different areas of the city to different computing entities, for example servers or smart sensors, all connected through a Fog Computing infrastructure.

Most parallel applications are designed so that the computation advances through successive steps, and all the nodes need to synchronize before proceeding to the computation related to the next step. For example, when using the master-slave model, the computation is "embarrassingly parallel"[5], i.e., the parallel tasks do not need to exchange data during the execution. When completing every step, however, the nodes must communicate the results to a central node that, after collecting all the data, gives the nodes the permission to execute the next step. Smart city applications differ from this model because they typically require that the computation regarding a specific region of the city is performed using the information received from a subset of neighbor regions. This corresponds to the necessity of synchronizing the computation only among a limited number of parallel nodes, without the need for a coordinator node.

Depending on the specific application, it is possible to tune the number of regions that partition the territory – and consequently their size – and the "synchronization degree", i.e., the number of neighbor regions that must communicate data among them and synchronize. The main objective of this paper is to show that the proper setting of these parameters is of paramount importance to balance the efficiency and effectiveness of computation. This is evaluated for the specific case of a "smart avenue" traversed by mobile devices held by vehicles and pedestrians, in which the goal of the computation is to predict the amount of internet traffic generated in each region of the avenue. The prediction of internet traffic is an important application scenario today [14][6], as numerous vehicles possess powerful sensing, networking, communication, and data processing capabilities, and can exchange information with each other (Vehicle to Vehicle, V2V) or exchange information 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)[7].

In particular, increasing the synchronization degree allows the accuracy of the prediction to be improved, because more devices are included in the computation, but can slow down the computation because of the larger overhead related to synchronization. Furthermore, increasing the degree of parallelization, i.e., the number of regions in which the avenue is partitioned, allows the computation to be fastened but generates the necessity of increasing the synchronization degree to keep the same accuracy, since each

region covers a smaller fraction of the avenue. The proper tradeoff between computation and accuracy should take into account the characteristics of the specific scenario, and can be formulated as an optimization problem with given constraints. For example, the system manager could be asked to maximize the accuracy of the computation given that it is completed within a given interval of time.

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 and the synchronization degree; finally, Section 5 concludes the paper.

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 this kind of scenario is simpler to model and the related results are easier to be analyzed. 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[7], smart devices can offer their computing and storage capabilities to perform computations in combination with the facilities of the fixed Cloud infrastructure. The goal of the smart avenue application is to predict the amount and characteristics of the data network traffic and the required computing and storage capabilities of devices in a future interval of time, starting from the past behavior of mobile devices. In this context, past behavior concerns both the usage of internet applications and the mobility behavior of the users. The accurate prediction of internet traffic can be used for several goals: to anticipate possible bottlenecks in some portions of the avenue, to save energy and batteries consumption by dynamically redistributing the workload between fixed and mobile devices, as recently described in [1], to design traffic-aware energy-efficient cellular networks [11], to improve the Quality of Service offered to the users, etc. To this aim, the use of machine learning algorithms for traffic forecasting through behavior modeling of mobile users is becoming a challenging issue to improve service effectiveness and efficiency [14][6]. For instance, usage pattern prediction of requests can be used to influence the admission/denial of service demands made by priority and non-priority users, in order to match their respective Quality of Service agreements [14]. As another example, the bandwidth provided in a given area can be dynamically adapted according to the predicted volume of requests, thus saving energy consumption in the overall network [6].

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", for example a smart device or a server. 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. For example, 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, i.e, 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 see that the number of nodes and the visibility radius are essential parameters to establish the proper tradeoff between computation time, speedup and accuracy of the result. The scenario of interest, outlined in Figure 1, is completed with the following assumptions:

- the length of the avenue under consideration is L which, unless otherwise stated, is set to 10 kilometers 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 time that would be needed by a single node to perform the overall computation for the entire avenue is T_{serial}, assumed to be equal to 10 minutes in the case that L=10 km;
- the computational load is uniformly distributed over the avenue, and the average time needed to perform the computation on a single node, T_{node}, is proportional to the length of the corresponding avenue portion, i.e., $T_{node} = T_{serial}/N$. The time is assumed to be distributed with a Gamma function with shape equal to 2. The variability can depend on many factors, among which the variable workload on the nodes and the variable number of involved devices.
- the time needed to communicate (transmit and receive) data with the neighborhood nodes is negligible with respect to the computation time. This assumption is coherent in the case that only summary and aggregated data are communicated, such as the number and type of mobile devices, the estimation about the global data that will be transmitted by such devices, etc.
- the mobile devices are assumed to 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 [14] and [6], but we use a simple model for two reasons: (i) it is sufficient to understand the basic behavior of the system; (ii) the analysis is not influenced and biased by additional assumptions that are often related to a specific domain or city.

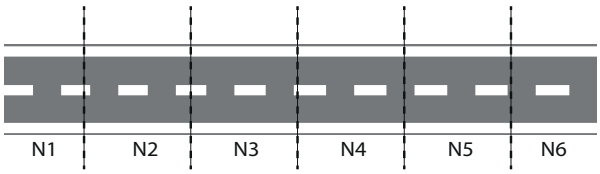

Fig. 1. Smart avenue scenario.

In this scenario, there is a clear tradeoff to achieve when setting the number of nodes N. 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 tradeoff concerns the value of R_V. On the one hand, a higher value of R_V is expected to slow down the computation, 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.

3 Petri net model for the computation

The parallel computation process for the described smart avenue application, and the synchronization barrier among the nodes, 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. Six Petri net *transitions*, labeled as N1–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[13], the transition is *enabled*, and the computation can start, if all the input places hold at least one *token*. When a transition fires (i.e., 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, i.e., 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 computation results to the neighbor nodes and the permission to such nodes to execute their computation at the next time step[1].

[1]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.

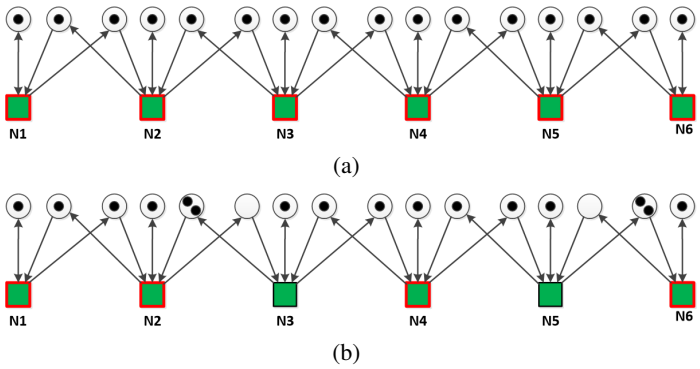

Fig. 2. Petri net representing the execution of tasks at six parallel nodes, with R_V equal to 1. In (a) all the nodes are ready to execute. After execution at nodes N3, N4 and N5, the state of the Petri net is depicted in (b): now N1, N2 N4 and N6 are ready to execute, while N3 and N5 must wait for the execution at nodes N2 and N6, respectively.

Figure 2(a) represents the state of the system in a situation where all the Petri net transitions are enabled, i.e., 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 2(b) 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. It is also noticed that 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 case of R_V equal to 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. 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 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

(e.g., 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. In the future, the nodes that are some steps behind can become faster and reach the other nodes, and so on. This is true if the assumption holds that the computation load is evenly distributed on the territory. If this does not hold, it is possible to divide the territory in a non-uniform fashion, for example, by assigning more nodes to the regions with the highest computational load. Overall, this allows the global computation to proceed faster, as will be shown in the next section, devoted to performance results. The results have been obtained in two ways: by using the well-known Petri net simulator Yasper[8], specifically its "automatic simulation" tool, and through an ad hoc simulator written in Matlab, which reproduces the same computation modeled by the Petri nets. Results are statistically identical, with the correlation factor always larger than 0.99.

4 Performance Results: Speedup and Accuracy of Computation

When setting the number of nodes N and the visibility radius R_V, a tradeoff emerges between minimizing the computation time and maximizing the accuracy of the computation. The next two subsections focus on these two aspects.

4.1 Computation time and speedup

In the first set of simulations, we examined 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, i.e., the visibility radius extends over the entire avenue. The average of the 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 time needed to execute a single step on all the nodes, T_{step}, by dividing the 30-days time interval by the number of completed steps[2]. The results are shown in Figure 3, where the all-to–all synchronization case is indicated with the label "All". It is seen that, with a given value of N, the computation is faster 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 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

[2] As explained in Section 3, there can be a gap between the steps executed at different nodes. Therefore, we consider the node that has executed the minimum number of steps.

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 value 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 3.

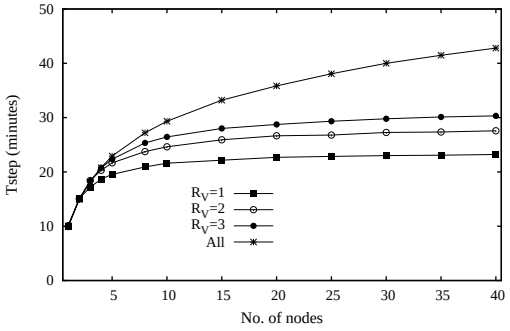

Fig. 3. Values of T_{step} vs. the number of regions, in the case that all the regions have the same length and the average computation time for a single region is 10 minutes.

After this first set of experiments, we performed a *strong scalability* evaluation[3], i.e., we focused on the case that the avenue, whose length L is fixed and set to 10 kilometers, is partitioned into a number of regions N. As described in Section 2, the average computation time at a single node, T_{node}, is proportional to the length of the region, l=L/N, and is equal to T_{serial}/N, where T_{serial} is assumed to be equal to 10 minutes. Figure 4 reports the values of T_{step}, the average time needed to perform a step an all the nodes. When N increases, the value of T_{step} decreases because the computation is partitioned among a larger number of nodes. Figure 5, reporting the speedup – i.e., the ratio between T_{serial} and T_{step} – is more useful to analyze the effect of the visibility radius R_V on the scalability. The effect is remarkable: as an example, when N is set to 40, with all–to–all synchronization the speedup is equal to about 9.3, while it increases to 13.2, to 14.5 and to 17.2 with values of R_V equal, respectively, to 3, 2 and 1. The corresponding speedup increments, in percentage, are 42%, 56% and 85%.

Ongoing experiments are showing that the speedup value greatly depends on the type of random distribution of the computation time, specifically on the coefficient of variation, i.e., the standard deviation/average ratio. With larger values of this ratio, the time needed for the synchronization increases, and the speedup decreases. Interestingly, however, we are also noticing that the improvement obtained when restricting the syn-

[3] 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

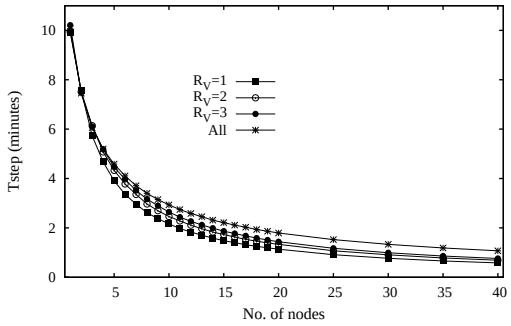

Fig. 4. Values of T_{step} in the case of an avenue with fixed length and partitioned among a variable number of regions.

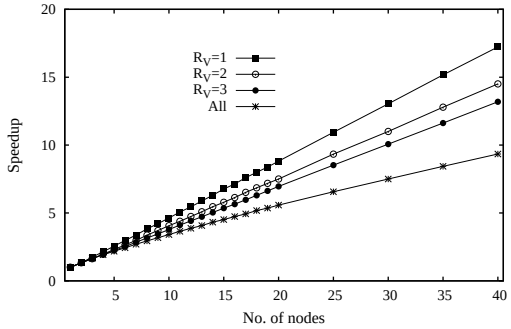

Fig. 5. Values of the speedup in the case of an avenue with fixed length and partitioned among a variable number of regions.

chronization to a few neighbor nodes (with respect to all–to–all synchronization) increases with the value of the coefficient of variation.

4.2 Accuracy of the computation

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, i.e., 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 ratio* C, or simply *coverage*, as the ratio between the number of visible regions and the number of regions from which mobile devices can arrive:

$$C = \frac{2 \times R_V}{N_R} \tag{1}$$

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

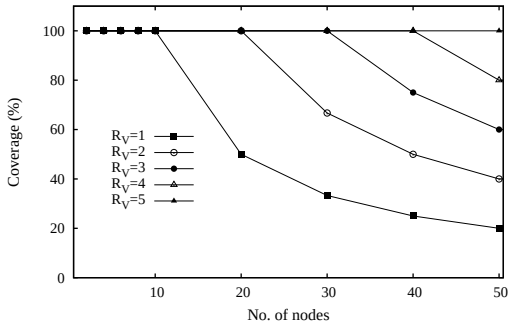

Fig. 6. Coverage ratio for mobile devices held by pedestrians.

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. Figures 6 and 7 show 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 kilometers 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 (1), is higher. In addition, it clearly appears that the coverage decreases with larger values of N and with smaller values of R_V. Figures 6 and 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 shows 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 frontier. Values of N and R_V that are not positioned on the frontier are not acceptable, because other choices of the parameter

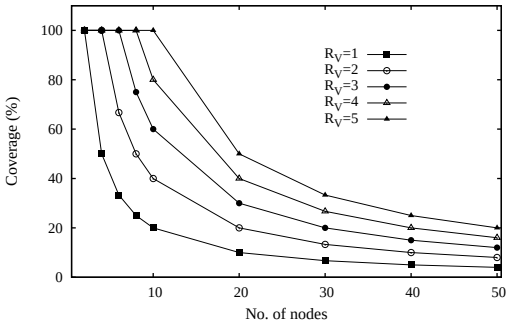

Fig. 7. Coverage ratio for mobile devices held by vehicles.

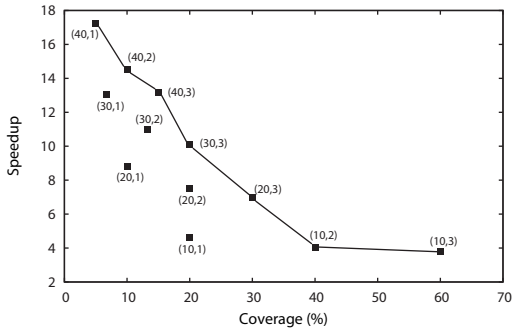

Fig. 8. Values of coverage and speedup for different values of the couple (N,R_V). The Pareto frontier is shown.

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.

5 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 predict 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. We showed that is possible to tune some system parameters – in particular, the number of parallel nodes and the number of neighbor regions among which the information is transmitted – to achieve the desired tradeoff between the accuracy of computation and its scalability and speedup. 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. Future work aims to investigate other use case scenarios, in which the computational load is not evenly distributed over the territory, or changes dynamically.

References

1. 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). pp. 549–553. Turku, Finland (2015)
2. 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. pp. 13–16 (2012)
3. Botta, A., de Donato, W., Persico, V., Pescapé, A.: Integration of cloud computing and internet of things: A survey. Future Generation Computer Systems 56, 684 – 700 (2016)
4. 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 (February 2016)
5. Ekanayake, J., Fox, G.: High performance parallel computing with clouds and cloud technologies. In: Cloud Computing, pp. 20–38. Springer (2010)
6. 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). pp. 120–129. Bologna, Italy (2013)
7. 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). pp. 1735–1739. San Jose, CA, USA (2013)
8. van Hee, K., Oanea, O., Post, R., Somers, L., an der Werf, J.M.v.: Yasper: A tool for workflow modeling and analysis. In: Proceedings of the Sixth International Conference on Application of Concurrency to System Design (ACSD '06). pp. 279–282. IEEE Computer Society, Washington, DC, USA (2006)
9. Krishnan, Y.N., Bhagwat, C.N., Utpat, A.P.: Fog computing- network based cloud computing. In: 2nd IEEE International Conference on Electronics and Communication Systems (ICECS). pp. 250–251 (2015)
10. Lee, I., Lee, K.: The internet of things (IoT): Applications, investments, and challenges for enterprises. Business Horizons 58(4), 431 – 440 (2015)
11. 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 Communications Magazine 52(6), 234–240 (2014)
12. Mitton, N., Papavassiliou, S., Puliafito, A., Trivedi, K.S.: Combining cloud and sensors in a smart city environment. EURASIP Journal on Wireless Communications and Networking 2012(1), 1–10 (2012)
13. Peterson, J.L.: Petri nets. ACM Comput. Surv. 9(3), 223–252 (September 1977)
14. Singh, R., Srinivasan, M., Murthy, C.: A learning based mobile user traffic characterization for efficient resource management in cellular networks. In: 12th Annual IEEE Consumer Communications and Networking Conference (CCNC). pp. 304–309 (January 2015)