Cluster Computing
https://doi.org/10.1007/s10586-019-02991-5

MAHA: Migration-based Adaptive Heuristic Algorithm for Large-scale Network Simulations

Muhammad Ibrahim[1] · Muhammad Azhar Iqbal[2] · Muhammad Aleem[3] · Muhammad Arshad Islam[3] · Nguyen-Son Vo[4]

Received: 29 May 2018 / Revised: 2 July 2019 / Accepted: 18 September 2019

Abstract
The scalable wireless network simulation poses huge computation challenges as the execution time needed to perform the simulation can be prohibitively high. Parallel and distributed simulation (PADS) approaches have been proposed that use huge memory and high processing power of multiple execution units [i.e., logical processes (LPs)] to handle scalable simulations. Each LP is comprised of a set of simulation entities (SEs) that can interact local or remote SEs. However, the remote communication among SEs and synchronization management across LPs are two main issues related to PADS execution of large-scale simulations. A number of migration techniques have been used to mitigate the problem of high-end remote communication. The problem is that most of the existing migration strategies result in higher number of migrations that ultimately lead to higher computation overhead. In this paper, we propose a migration-based adaptive heuristic algorithm (MAHA). Considering the run-time dynamics of the wireless network simulations, MAHA provides dynamic partitioning of the simulation model to achieve better local communication ratio (LCR). In addition, an adaptive academic simulation cloud platform, namely *A-SIM-Cumulus* cloud, is deployed for scalable simulations. The MAHA is implemented on A-SIM-Cumulus Cloud and simulations are executed multiple times with different configurations and execution environments. The results with optimum LCR show that the proposed algorithm significantly reduces the number of migrations and achieves a good speedup in terms of parallel (i.e., both multi-core and distributed) execution.

Keywords MAHA · EHA · A-SIM-Cumulus · Large-scale wireless network simulation · Migration · ARTIS/GAIA

1 Introduction

From the past few decades, network simulators are reckoned as eminent recourse for network researchers. The complex nature, expensive experimentations, and limitations of conventional analytical evaluation methods adhere the espousal of network simulation [1, 2] for pre-deployment performance evaluation of communication and wireless network systems [3]. The most common approach for network simulations is based on monolithic design, where a single execution unit is in-charge of managing all parts of the simulation. The monolithic approach is more realistic for small-to-medium scale network simulations [4]. On the other hand, scalable wireless network simulations with microscopic details are impractical to execute on monolithic systems. Generally, microscopic details of a wireless network simulation model ascertain the accuracy of simulation results and are essential for the successful deployment of the proposed model. Concerning the monolithic systems, memory scarcity and low computing power are two important factors that limit the scale of network simulations [3]. This is due to the facts that (1) memory requirements increase exponentially (as

✉ Muhammad Azhar Iqbal
m.a.iqbal@swjtu.edu.cn

1 Department of Computer Science, Virtual University of Pakistan, Rawalpindi 46000, Pakistan

2 Department of Computer Science, SWJTU-Leeds Joint School, Southwest Jiaotong University (SWJTU), Chengdu 611756, People's Republic of China

3 Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan

4 Institute of Fundamental and Applied Sciences, Duy Tan University, Ho Chi Minh City 700000, Vietnam

representation of each node's state requires a certain amount of memory) and simulations takes long time span to be completed and (2) numerous interactions (i.e., event, packets) are required to be created, stored and delivered instantly [4, 5].

As an alternate, parallel and distributed simulation (PADS) [6] approach effectively executes the large-scale simulations. In PADS, the simulation is segregated into different parts and each part is assigned to specific logical unit known as logical processor (LP). These LPs may reside on same or different physical execution unit (PEU). The PADS has a capability to employ a huge amount of memory and computation resources from different PEUs. For precise simulation execution using PADS, all LPs are required to be synchronized during the whole span of a simulation. PADS simulations could be implemented on multiple platforms, such as grid computing [7], high performance computing (HPC) [8], Cloud computing [9] etc. In recent advancements, Cloud computing eliminates the barriers of resource limitations for large-scale applications [10]. The implementation of PADS over Cloud computing provides an essential platform to address the problems encompassing scalable simulations with microscopic details. The proposed academic Cloud leads to good performance in terms of simulation execution time. The initial employed partitioning scheme was based on static allocation of simulation entities (SEs) on a number of LPs. The goal of this scheme was two folds: the load on different LPs should be balanced keeping the remote communication (among SEs located on different LPs [11]) as minimum as possible. However, the static partitioning mechanism embroils two issues that are: latency overhead of remote communication (inter-LP communication for parallel execution) and synchronization management among the LPs.

In this paper, we are considering large-scale wireless network simulations. The nodes in the wireless networks are mobile in nature and frequently move due to high mobility involved. This leads to high-end remote communication and thus every form of static partitioning may not be an optimal choice. The partitioning problem becomes more complex due to a number of factors that must be taken into account. The LPs in the simulation may not be homogeneous [5] in nature and the communication pattern may vary during the course of the simulation as the message sending by the SEs is random in wireless networks. Moreover, the execution architecture may be comprised of several components that can be heterogeneous (i.e., both in terms of hardware used and performance). In contrast to the static partitioning techniques, the dynamic partitioning was proposed to keep the load balanced on LPs and reduce high-end remote communication [12]. The dynamic partitioning techniques use the concept of SE migrations from one LP to another with the aim to reducing the high remote communication. The migration is triggered based on the condition specified by simple heuristics that are evaluated for each SE at their respective LPs. The local available information is used for the migration decision. The migration mechanism is an attempt to localize the communication among the SEs as much as possible and to keep the load balanced on different LPs. The migration mechanism has a certain computation cost which is crucial for achieving good performance. The migration approach will be effective if the cost associated with employing the migration is lower than the remote communication cost. Another important consideration is related to the number of migrations which is required to minimize the remote communication among the SEs located on different LPs. The required number of migrations raises frequently due to the continuous node mobility in wireless networks. Thus, more computing and memory will be required to achieve optimal LCR (local communication ratio). The LCR corresponds to the ratio of the total number of local communications by all SEs in a given LP, with respect to all interactions originating from this LP [5]. The proposed approaches [5, 9, 12] are appropriate to handle the networks with no or very low mobility nodes.

The high mobility involved in mobile wireless networks leads to more frequent migrations which attribute huge migration cost, ultimately leading to poor simulation performance. To address the migration related issue in this research, we present migration-based adaptive heuristic algorithm (MAHA). MAHA considers the transitive dependency communication property of the SEs. The proposed algorithm provides a dynamic partitioning of the simulation model based on runtime requirements of the wireless network simulations. The algorithm uses an adaptive heuristic for migration decision in order to reduce the number of migrations with an ultimate goal to achieve better LCR. The proposed algorithm is better in terms of achieving optimum LCR with a reduced number of migrations. This leads to a reduction in the total time needed to execute the simulation. In addition, an adaptive academic cloud namely A-SIM-Cumulus is proposed and implemented for large-scale wireless network simulations. A-SIM-Cumulus framework integrates ARTIS (advanced run-time infrastructure)/GAIA (generic adaptive interaction architecture) middleware [13] with our implemented academic Cloud [14].

The rest of this paper is organized as follows: Sect. 2 describes the related work about partitioning and migration approaches for PADS. In sect. 3, we discuss the components of proposed adaptive academic Cloud framework namely A-SIM-Cumulus. Section 4 provides the details regarding migration mechanism and proposed heuristics. Section 5 presents the simulation setup and results discussion. Finally, Sect. 6 reports our conclusions and future work.

2 Related work

Executing large-scale wireless network simulations in parallel involves partitioning of the simulation model into a number of smaller components also called LPs [5]. The components encompass part of the simulation that need to be executed independently on an individual PEU. Concerning the partitioning of simulation models in a distributed environment, a number of approaches have been proposed to maintain a global shared-state in the distributed simulation. Some partition schemes are based on the static approach [15], while other schemes such as simulation domains [16], data distribution management [17, 18], dynamic adaptive partitioning [19], and hierarchical federations [20] use dynamic approach. The authors presented a static partitioning approach for parallel conservative simulations in [21]. In [22], the authors proposed the static as well as dynamic load balancing strategies. The proposed approach relies on the use of conservative synchronization algorithm and thus is applicable only on shared memory architecture. In [23], the authors introduced a novel approach for the partitioning of disjoint parts of the large-scale network topology and assigning them to different LPs for performance efficiency. Three set of partitioning strategies (i.e., random, k-cluster, and METIS) are implemented in order to quantify the performance of the proposed partitioning scheme. The authors in [24] proposed a novel partitioning technique that uses the concept of convex hulls for partitioning of crowded simulations. The authors in [24–27] presented the work related to the provisioning of the Internet of Things (IoT) simulations. The proposed methodology in [25] integrates multiple simulation models (i.e., OMNeT++, MATLAB, and ARTIS/GAIA) for the implementation of IoT-based complex scenarios. Logan et al. [28] floated a novel agent-based approach for PADS simulations. In their approach, each of the agent is implemented as distinct LP. The dynamic partitioning method is used to implement the parallel simulation using shared state approach. The proposed approach supports conservative as well as optimistic synchronization in an adaptive way to meet the requirements of computation and communication demands of the simulated model. The associated cost related to the migration is not optimal and may vary due to the involvement of high mobility nodes. Most of the PADS solutions have been suffered from high cost of remote communication and computation required for synchronization management. A number of techniques have been proposed to accomplish the speedup in simulations using migration and load-balancing for PADS simulation [5, 9, 12]. In [29] and [30], the authors have proposed dynamic partitioning algorithm for optimistic distributed simulation based on the migration of parts of the simulated model. The proposed approach reduces the computation and communication cost using migration and load-balancing mechanisms. Anglo et al. [30] have introduced an adaptive and dynamic approach to obtain simulation speedup through the use of two proposed heuristics for efficient migration and load-balancing of SEs. In [18], the authors have proposed ARTIS Middleware, which supports PADS for large-scale wireless networks over heterogeneous execution architecture. The ARTIS uses migration mechanism to minimize the remote communication among SEs located on different LPs. Moreover, ARTIS is in combination with GAIA [31] is implemented for load-balancing of SEs across different LPs. Anglo et al. [32] have designed a new simulation tool namely *PaScas* that enables the simulation of scale-free networks. The ARTIS middleware is used as the underlying module for remote communication provisioning and management among SEs (i.e., located on different LPs). In addition, the authors have presented a number of algorithms (named as GOSSIP) as test cases to evaluate the performance of the scale-free networks. The proposed solution leads to a good level of LCR; however, the number of migrations leads to an extra overhead. Thus, it produces a negative impact on the overall performance gain. The author in [33] presented self-scalable distributed network simulation environment for Cloud computing. A slightly different technique has been proposed in [34] that uses a fault-tolerant parallel simulation Middleware to support PADS. The SEs are replicated and distributed on different LPs, which consequently provide a mechanism to handle crash-failures of execution units. In [5], PADS approach was presented that used the concept of dynamic partitioning (i.e., migration) of the simulation model to handle large-scale wireless network simulation. A set of heuristics are being employed that are evaluated (i.e., locally on each LP) for the SEs migration during the simulation execution.

Conclusively, most of the existing proposed approaches have certain deficiencies that must be mitigated to ensure the smooth simulation of PADS on Cloud architecture. In particular, there are two issues as given below. First, most of the schemes have improved the LCR; however, the increased number of migrations ultimately leads to the migration cost. Second, the proposed approaches are appropriate to handle the networks with no or very low mobility nodes. However, the existing approaches degrade the performance for the networks (i.e., wireless networks) involving high mobility nodes. High mobility in these networks will lead to more frequent migration causing a huge migration cost, ultimately leading towards poor simulation performance. To overcome these issues, we present MAHA algorithm that uses the transitive dependency communication property of the SEs. MAHA provides a dynamic partitioning of the simulation model based on runtime requirements of the wireless network simulations. The algorithm uses an intelligent heuristic for migration decision in order to reduce the number of migrations with an ultimate goal to achieve better LCR. The proposed algorithm is better in terms of achieving optimum LCR with reduced number of migrations. This leads to a reduction in the total time needed to execute the simulation.

3 Adaptive SIM-Cumulus architecture

The A-SIM-Cumulus comprises of four layers i.e., System Accessibility Layer (SAL), Cloud Instance Management Layer (CIML), Virtual Platform Layer (VPL), and Physical Infrastructure Layer (PIL) as shown in Fig. 1. The details of necessary A-SIM-Cumulus components required to perform network simulations are presented below.

3.1 System accessibility layer (SAL)

At the front end, the SAL enables users to interact with cloud instance management layer (CIML) through the use of cloud front-end interface (CFI) and remote desktop protocol (RDP). At CFI, REpresentational state transfer (REST) API (web services interface) is used to provide and acquire registration and usage information, respectively.

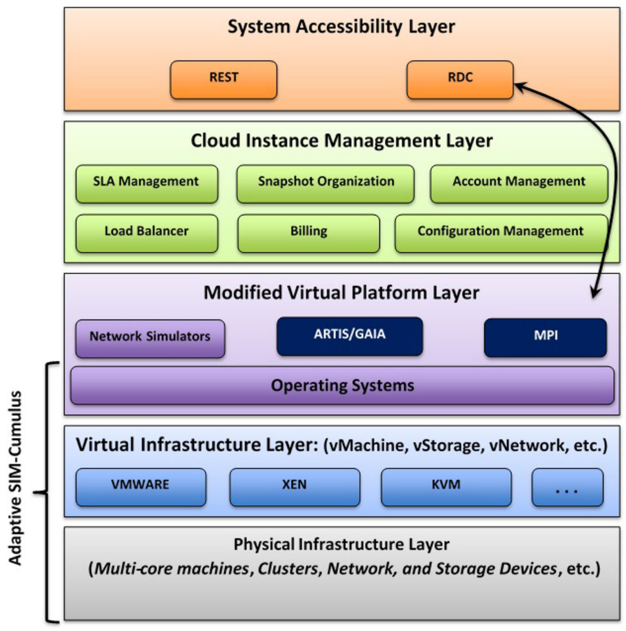
System Accessibility Layer
REST
RDC
Cloud Instance Management Layer
SLA Management
Snapshot Organization
Account Management
Load Balancer
Billing
Configuration Management
Modified Virtual Platform Layer
Network Simulators
ARTIS/GAIA
MPI
Operating Systems
Virtual Infrastructure Layer: (vMachine, vStorage, vNetwork, etc.)
VMWARE
XEN
KVM
. . .
Physical Infrastructure Layer
(Multi-core machines, Clusters, Network, and Storage Devices, etc.)
Adaptive SIM-Cumulus

Fig. 1 A-SIM-Cumulus architecture

RESTful web services allow end-users to configure (i.e., selection of operating system, a network simulator, mobility model, map, and traffic generators etc.) and launch the required instances on the A-SIM-Cumulus cloud. The RDP connection manager is used to launch the instance of A-SIM-Cumulus to perform the required simulations.

3.2 Cloud instance management layer (CIML)

The CIML is comprised of two components that are: cloud instance resource management (CIRM) and Cloud provenance. The core responsibility of CIRM is to configure and manage Cloud instances of the SIM-Cumulus. CIML hides the complexities associated with configuration setup of the simulation tools from users. The description of other related components of the adaptive academic Cloud A-SIM-Cumulus can be found in [14].

3.3 Modified virtual platform layer (MVPL)

The MVPL is responsible for the provisioning of Cloud instances configured with various network simulators (i.e., MATLAB, NS2, OMNeT++ etc.). The MVPL uses the Eucalyptus services and enable users to access Cloud instances. ARTIS and GAIA are integrated into VPL layer to perform parallel simulations [14]. The ARTIS is middleware that implements several modules to optimize parallel simulation (of microscopic, dynamic, and complex nature systems) over homogeneous and heterogeneous execution platforms [31]. The ARTIS offers several services such as simulation bootstrap and termination, LP coordination, track runtime statistics, synchronization management, and managing the interaction primitives for the communication among SEs on different LPs [5]. The GAIA framework allows the migration of SEs from one LP to another LP. The migration decision in GAIA is implemented on the basis of heuristics that basically involves the local and remote communication among different SEs. The local and remote communication of the SEs is useful in determining the migration decision of a particular SE. If the ratio of remote to local communication of an SE exceeds a certain threshold, then that SE is considered as candidates for migration. The migration has a certain computational cost, which could be very crucial. In other words, the introduction of the complex heuristic algorithm for migration decision may attribute to a high overhead. Moreover, the heuristic may contribute to higher number of migrations while achieving the same LCR value. This work exploits the migration and communication services of ARTIS and GAIA middleware to perform adaptive migration of SEs dynamically. In this paper, the MAHA is proposed to use an intelligent heuristic for migration decision and minimize the number of migrations with an ultimate goal to achieve better LCR. The objective is to pay some extra computation with the aim to reduce the computation overhead required for large number of migrations. The MAHA algorithm is better in terms of achieving the same LCR with reduced number of migrations. The proposed algorithm is implemented and evaluated using our proposed Cloud framework known as A-SIM-Cumulus. The details pertaining to migration decision is provided in Sect. 4.

3.4 Virtual infrastructure layer (VIL)

The VIL resides at the lower layer of A-SIM-Cumulus that exposes the Cloud resources to the upper layers. A-SIM-Cumulus uses *Eucalyptus* Open Source Cloud platform [35] architecture. The *Eucalyptus* supports private as well hybrid Cloud implementation [36]. Moreover, Eucalyptus is inherently flexible and interoperable with most of the advanced commercial Clouds (i.e., Amazon EC2, IBM SmartCloud etc.). KVM is used as a baseline hypervisor for virtualization of A-SIM-Cumulus Cloud. In addition, live migration of running VMs is also supported by KVM.

3.5 Physical infrastructure layer (PIL)

The PIL deals with the physical computing resources such as: *multi/many-core machines, clusters, data-centers, networks, storage devices* etc. All the resources are pooled and provided to the upper layer in an on-demand fashion.

4 MAHA: migration-based adaptive heuristic algorithm

The PADS provides the opportunity to solve simulation problems that encompasses massive nodes and micro-level details. The PADS simulation requires the partitioning of simulation model into a number of equal partitions and then places each component on a separate execution component called LP. Each LP is responsible for the management and handling of simulation events among the SEs that are located on it. A major hurdle in getting the required gain in performance is due to the LP synchronization and remote communication cost in the PADS simulation [5]. Moreover, the distributed simulation environment is dynamic and unpredictable in nature with respect to the network load. Partitioning the simulation model in an effective way can attribute to the performance gain. However, the static partitioning of wireless networks simulation does not help in a performance gain due to involvement of the high mobility which could lead to an increase in the cost of LP synchronization and remote communication. Thus, every form of static partitioning is inadequate and therefore, requires a dynamic and adaptive migration approach. In this work, we aim to propose an adaptive migration algorithm that automatically reconfigures the simulation in an adaptive fashion to meet the runtime dynamics. This is the starting point to design and implement a mechanism that will result in a reduction in communication cost and is discussed in Sect. 4.1. The proposed adaptive heuristic and algorithm details are presented in Sects. 4.2 and 4.3, respectively.

4.1 Remote communication cost reduction

The simulation model comprises a number of SEs which interact with each other during the simulation execution. The intra-LP communication between SEs has low link latency. However, the inter-LP communication between SEs may face inconsistent latency depending on the type of link between the LPs. Figure 2 shows the intra-LP and inter-LP communication of different SEs. The main idea is to observe the communication pattern of every SE during simulation execution and decides whether the migration of SE to another LP is necessary. The SEs that have high interaction with other remote SEs during certain time period are considered as appropriate candidates for migration. In this way, the high cost of inter-LP (remote) communication is reduced. The migration leads to a performance gain, however; it may results in performance degradation if the cost of migration is higher than remote communication cost.

4.2 Heuristic basis

In this research, we consider WLAN simulation to evaluate the performance of the proposed adaptive heuristic algorithm for large-scale PADS. The wireless communication networks are suitable as the interaction among the wireless nodes is generally location dependent. In wireless networks, all the nodes move due to the mobility changes which lead to a change of the neighbor nodes and network topology. Therefore, the communication pattern among the SEs changes very frequently during the course of the simulation.The SEs may follow a communication pattern, where a transitive dependency among SEs may exist. For example, existance of transitive dependency among three different SEs (as shown in Fig. 3), including SE_2, SE_4, and SE_{10} that are located in LP_1, LP_2, and LP_3, respectively.

It is considered that SE_2 has most of its interaction with SE_4 and at the same time SE_4 had high interaction with SE_{10}. Thus, after a certain amount of time, SE_2 needs to be migrated to LP_2 and SE_4 needs to be migrated to LP_3. In order to cluster SE_2 with SE_4, the SE_2 is first migrated to LP_2 and then moves to LP_3. This will lead to extra

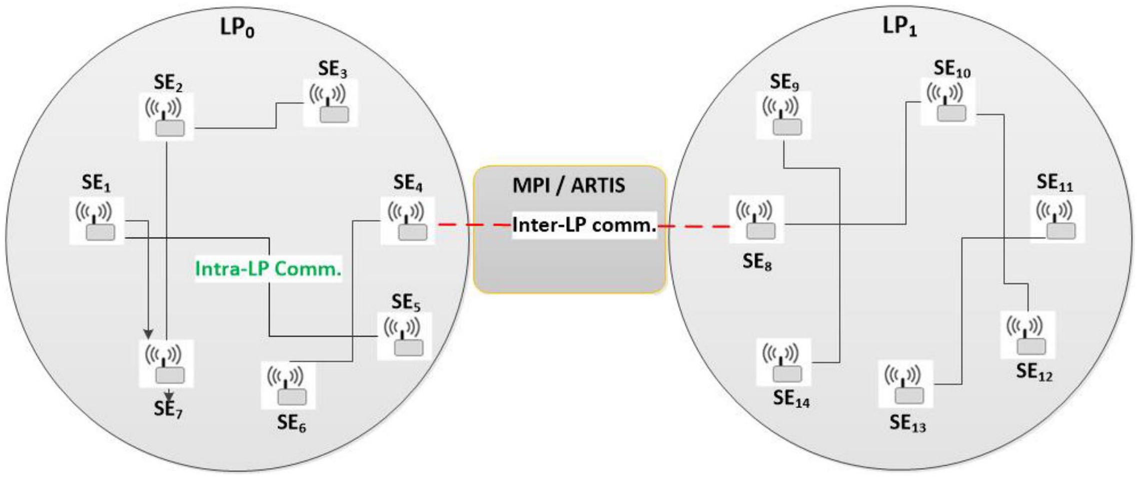

Fig. 2 Intra-LP and Inter-LP communication between SEs

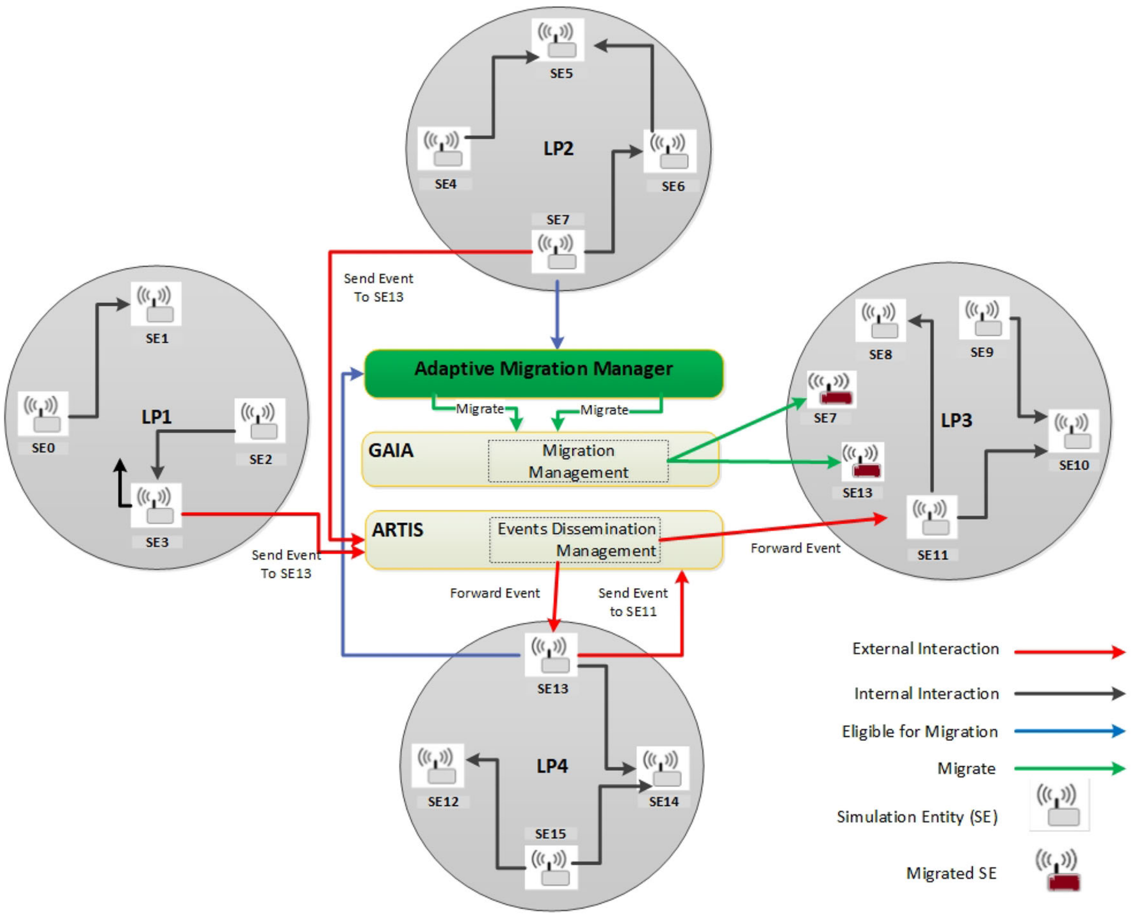

Fig. 3 SE internal/external interaction and migration execution

overhead and resources in moving SE_2 to LP_2 first and then migrating to LP_3. The problem is more specifically known as a transitive dependency. To resolve this issue, this work proposes an adaptive heuristic algorithm designed to handle migrations in PADS environment. One of the evaluation approaches is to execute the heuristic evaluation at each timestep [5]. This evaluation is beneficial for systems that involves a lot of communication among the SEs and there are rare chances that a timestep results in zero interaction. However, with a large number of SEs the cost of heuristic evaluation can lead to extensive computation, that will, in turn, result in performance degradation. The alternate approach for heuristic evaluation uses the interaction size as a decision parameter for evaluation. The evaluation is triggered only if an SE has sent at least M messages since the last evaluation. The value of M represents the maximum size of interaction window. In many large-scale systems, this evaluation approach greatly reduces the number of evaluation at each timestep and thus leads to more scalable performance. The base condition used for the evaluation of this approach employs the technique introduced in [5]. The algorithm checks whether there exist some SEs that meet the condition of the transitive dependency and thus migration is triggered.

4.3 Migration-based adaptive heuristic algorithm

The proposed MAHA algorithm uses the adaptive heuristic for the implementation of migration during the simulation. MAHA is comprised of four algorithms in order to perform SE migrations to appropriate LPs. In Algorithm 1 (lines 1–2), the values of CommWinSize and MaxWinThreshold are initialized. The CommWinSize and MaxWinThreshold parameters are used in the decision-making pertaining to the short-listing of SEs for migration evaluation. Next, all SEs eligible for migration during each timestep is determined by the algorithm (lines 3–14). During migration eligibility computation phase, the local and remote communications of a particular SE interaction are added to calculate the total interactions of that SE at each timestep (lines 6–7). The CommWinSize is determined on the basis of total *SEInteractions* (line 8). At line 9, all SEs that have done sufficient communication are evaluated for the eligibility of possible migration. The information about SE local interaction, remote interaction, and LP are added to hash table SEMigEvalHashtbl (line 10). The hash table is then passed to EvalSEforMig module (in Algorithm 2), which is used to find out all SEs that need migration.

Algorithm 1 FindSEMigEval - Finds all SEs Eligible for Migration Evaluation

```
Input: $S_{SE} = ListofSEs(i.e., SE_1, SE_2, SE_3, \ldots, SE_n)$
Output: $SEMigEval_{Hashtbl}$

1:  $CommWinSize \leftarrow 0$
2:  $MaxWinThreshold \leftarrow N$
3:  for all $t\ timesteps$ do
4:      for all $LPs$ do
5:          for all $SEs\ in\ the\ communication$ do
6:              $SE_{Interaction} += SE_{LocInt}$
7:              $SE_{Interaction} += SE_{RemInt}$
8:              $CommWinSize += SEInteraction$
9:              if $(CommWinSize >= MaxWinThreshold)$ then
10:                 $Add\ SE_j,\ LP_i,\ SE_{LocInt},\ SE_{RemInt}\ to\ SEMigEval_{Hashtbl}$
11:             end if
12:         end for
13:     end for
14: end for
15: EvalSEforMig($SEMigEval_{Hashtbl}$)
16: Return $SEMigEval_{Hashtbl}$
```

The Algorithm 2 determines all the SEs that need to be migrated upon their eligibility. The information regarding the local interaction, remote interaction and LP (i.e., where the SE is located) is provided as an input to Algorithm 2 in the form of SEMigEvalHashtbl. The flaglist, Migration Threshold (migTS), Standard Migration Factor (SMF), and Migration Factor (Migf) are initialized (lines 1–4). The SEs that are eligible for the migration are flagged and necessary details required for migration are added to hash table MigHashTbl (lines 5–11). The Migration Factor (MF) is calculated using SElocInt and SEremInt of the SE (i.e., MigF = SERemInt / SELocInt) as given in line 6. After the determining of MigF, the SE is considered for migration only (line 7) if: MigF > SMF and at least ΔTS_{SE} timesteps having passed since the last evaluation of the corresponding SE.

All the SEs that meets the criteria mentioned at line 7 are added to flagList (line 8). The information regarding source SE, Destination SE, Source LP, and Destination LP is added to hash table MigHashTbl (line 9). This information is given as an input to FlagMig module given in Algorithm 3. All the SEs that meets the criteria mentioned at line 7 are added to flagList (line 8). The information regarding Source SE, Destination SE, Source LP, and Destination LP is added to hash table MigHashTbl (line 9). This information is given as an input to FlagMig module given in Algorithm 3.

Algorithm 2 EvalSEforMig - Finds all SEs to be migrated

```
Input: $MigEvalHashtbl, SMF, MigF, TS_{SEi}, migTS$
Output: $MigHashTbl$

1: $flagList[]$
2: $migTS \leftarrow 10$
3: $SMF \leftarrow 1 - 20$
4: $MigF \leftarrow 0$
5: for all $i$ in $SEMigEval_{Hashtbl}$ do
6:     $MigF = SE_{RemInt} / SE_{LocInt}$
7:     if ( $MigF >= SMF$ AND $\Delta TS_{SEi} > migValue$) then
8:         $flagList[i] = SEi$
9:         $Add$ $src_{SE}, dest_{SE}, src_{LP}, dest_{LP}$ $toMigHashTbl$
10:    end if
11: end for
12: FlagMig(MigHashTbl)
13: Return $MigHashTbl$
```

Algorithm 3 FlagMig - Filter SEs based on transitive dependency to be flagged for migration using MAHA approach.

```
Input: $MigHashTbl$
Output: $SEsMigList$

1: $i \leftarrow 0$
2: $j \leftarrow 0$
3: for all $i$ in $MigHashTbl$ do
4:     for all $j$ in $MigHashTbl$ do
5:         if ($destSEi == srcSEj$ ) then
6:             $flagList[i] = SEi$
7:             $execMig(destSE[i], destSE[j], MigHashTbl)$
8:             $execMig(srcSE[j], destSE[j], MigHashTbl)$
9:         else
10:            $ExecMig(srcSE[i], destSE[i], MigHashTbl)$
11:        end if
12:    end for
13: end for
14: Return $SEsMigList$
```

The first two algorithms represent the EHA [5] approach which is extended by proposed MAHA in Algorithm 3 and 4. Algorithm 3 represents the MAHA core algorithm to determine all those SEs that meet the criterion of transitive dependency required for migration decision. The input of Algorithm 3 is the MigHashTbl containing information regarding Source SE, Destination SE, Source LP, and Destination LP. All the SEs meeting the criterion of transitive dependency are flagged for migration and added to MigHashTbl table (lines 1–11). Each destination SE is compared with all the source SEs one by one and if the Destination SE is Source SE in another record, then both Destination SEi and Source SEj are migrated to LP where destination SEj is located (line 5–8).

Algorithm 4 ExecMig - Filter SEs are migrated to the respective LP.

```
Input: $SEsMigList$
Output: $FlagedSEsMigrated$

1: $k \leftarrow FlagedSEs$
2: for all $k$ in $SEsMigList$ do
3:     $MigrateSEs(srcSE[k], destSE[k], MigHashTbl)$
4: end for
5: Return $MigratedSEsList$
```

4.4 Partitioning mathematical model

The ARTIS/GAIA as Parallel Discrete Event Simulation (PDES) [37] provides flexible platform for the execution of large-scale simulation. The simulation speedup using PDES can be achieved by distributing the simulation over a number of LPs that run in parallel. Each LP maintains their simulation clock independently and is responsible to keep track of the concurrent events execution [38]. ARTIS framework is utilized to perform communication and synchronization management among multiple LPs. To support dynamic partitioning of the simulation model based on the run-time dynamics of the simulation, the GAIA framework is used. The SEs that have high remote communication are analyzed during communication and migration is executed if required to minimize the remote communication. The detailed mathematical modeling of static and dynamic (i.e., SEs migration) partitioning is discussed as below.

Let α represents the set of available LPs then

$$\alpha = \{LP_1, LP_2, LP_3, \ldots, LP_N\} \tag{1}$$

and β be the set of SEs in the simulation model:

$$\beta = \{SE_1, SE_2, SE_3, \ldots, SE_n\} \tag{2}$$

Initially, the Simulation model is divided into equal number of static partitions i.e., LPs and each is assigned equal number of SEs.

$$\gamma = n/N \tag{3}$$

where **n** represents the number of SEs in the simulation and **N** represents the number of LPs. In order to execute parallel simulation, SEs are required to be distributed across different LPs. The allocation of specific range of SEs on different LPs can be obtained by the given equation:

$$LP_i = \{(i-1)(N)+1, (i-1)(N)+2, \ldots, (i-1)(N)+N\} \tag{4}$$

To execute simulation in parallel, SEs are required to be distributed across different LPs. The simulation execution is started after the initial placement of SEs to their respective LPs. During the simulation execution, SEs communicate locally within the same LP or remotely with SEs located on other LP. The information about local and remote communication is maintained by each of the LP locally and if the remote communication crosses a certain threshold the migration will be required to minimize the high-end remote communication. Let Γ represents the ratio of the remote communication to the local communication then

$$\Gamma = SE_{RemInt}/SE_{LocInt} \tag{5}$$

The migration will be required only if:

$$\Gamma > \omega \tag{6}$$

and at least ρ timestep has passed since last migration. ω is standard migration factor used to control the number of migrations. This represents the EHA approach employed for migration execution. Although, the obtained local communication is increased; however, the number of migration tends to increase for wireless network with high mobility nodes. The MAHA approach is employed to control the number of migrations achieving an approximate same level of LCR. All SEs that were tagged for migrations are given as an input to the MAHA. The SEs will be finalized for migration only if the SEs meet the transitive dependency property (psudocode shown below).

```
for all $i\ in\ MigHashTbl$ do
    for all $j\ in\ MigHashTbl$ do
        if $(destSEi == srcSEj)$ then
            $flagList[i] = SEi$
            $execMig(destSE[i], destSE[j], MigHashTbl)$
            $execMig(srcSE[j], destSE[j], MigHashTbl)$
        else
            $execMig(srcSE[i], destSE[i], MigHashTbl)$
        end if
    end for
end for
```

4.5 Cost analysis

We now discuss the cost associated with the simulation execution in sequential/parallel mode. The overall execution cost (OEC) is the time needed to complete the simulation execution. OEC contains two costs that are (1) state updating cost (SUC) and (2) local interaction cost (LIC).

$$OEC = SUC + LIC \tag{7}$$

The SUC cost is associated with updating the state variables during each simulation event whereas the LIC is cost involved local communication among SEs. The OEC for parallel simulation execution is obtained as follows:

$$OEC = SUC + GCC \tag{8}$$

The generic communication cost (GCC) is comprised of interaction cost (IC), middleware management (MM), and synchronization cost (SynC).

$$OEC = SUC + (IC + SynC + MM) \tag{9}$$

The IC is the cost associated with message delivery among the SEs. This cost depends on a number of factors that are the size of message and the location (LP) of the receiving SE. The location is important as it can lead to a subtle difference in cost if the SE are located locally. The IC can be either local and remote interaction cost. The LIC refers to the cost involved in delivering the message to the SE on a local LP and Remote Interaction Cost (RIC) refers to the SE communication with a remote SE as shown in Equation 10.

$$IC = LIC + RIC \tag{10}$$

Thus, OEC can be calculated as

$$OEC = SUC + (LIC + RIC + SynC + MM) \tag{11}$$

Performance depends on the ratio between local and remote communications. Let L_{Comm} represents the size of local communication and R_{Comm} represents size of remote communication then LR_R (i.e., local to remote communication ratio) can be obtained by

$$LR_R = L_{Comm} \mid R_{Comm} \tag{12}$$

An increase in the LR_R corresponds the performance improvement in simulation time.

Given the cost difference between these two communication types, the best performance can be obtained by maximizing the local interactions. For this reason, we propose a migration mechanism that re-allocates SEs among the LPs, i.e., a mechanism that changes the partitioning configuration. This aims to cluster the SEs that interacts frequently in the same LP, reducing the use of costly inter-LP communications. Taking into account all of these aspects, we have

$$OEC = SUC + (LIC + RIC + SynC + MM) + Mig_C \tag{13}$$

where Mig_C is the migration cost. It means that the total execution cost of the PADS now includes the computation and communication costs paid for the reallocation of SEs. More in detail, Mig_C can be written as.

$$Mig_C = Mig_{CPU} + Mig_{Comm} + Heu_C \tag{14}$$

where Mig_{CPU} is the computation cost involved in executing the migration. The Mig_{Comm} refers to cost involved in moving an SE from one LP to the remote LP upon migration and Heu_C is the computation cost associated with the migration evaluation.

5 Performance evaluation

For simulation experiments, we consider large-scale wireless networks with microscopic details and high-density nodes. The computational complexity of the network simulation is magnified in wireless networks due to the involvement of continuous node (i.e., SE) mobility; especially in the case of large-scale deployment [39]. The choice of the model is Agent-based approach [5], where the agents correspond to the simulation entities. The simulation area comprises of two dimensions wherein each agent is allowed to move freely. All the agents communicate with their neighbor nodes that are located in their proximity. In other words, if any SE has a new interaction, it will be forwarded to all SEs that are situated within a threshold distance. To comply with the interaction requirements, the Random Waypoint (RWP) mobility model is selected [40]. It is one of the most prevalent mobility models that allows unrestricted movement of all of the agents in the entire simulated area and these agents are not correlated with each other. To obtain the simulation results, the experiments are performed using the proposed algorithm and EHA [5]. Three sets of simulation experiments are executed to scrutinize the performance of the adaptive algorithm.

5.1 High migration effects on obtained LCR

In the first round of simulation experiments, a total of 10,000 SEs are distributed on eight LPs. The SEs can send an interaction to other SEs that are located within a range of 250 space units. Each SE is placed in random positions according to two dimensional plane and is assigned to a certain LP. An equal number of SEs are placed in each LP and the assignment of SE on a certain LP is performed randomly. Each simulation run is executed for 3600 seconds, occupying the 10,000 × 10,000 space units of simulation area. The probability of interaction (where each of the SE can communicate at a given timestep, during the simulation) is set to 0.5. This indicates that during a certain timestep, half of the SEs are allowed to send messages. The mobility model is *RWP* and the mobility speed is in the range of 1–25 spaceuits per timestep. The range of MF (Migration Factor) is set to 1–19 and the Migration Threshold (MT) is equal to 10.

For every MT value and mobility speed value in the specified range, an independent simulation is executed. Figure 4 plots the relationship between the total number of migrations, the obtained LCR and the mobility speed. The results reveal that for low-speed mobility values, a limited number of migrations can lead to a very high LCR values. The important observation is that for a static allocation of

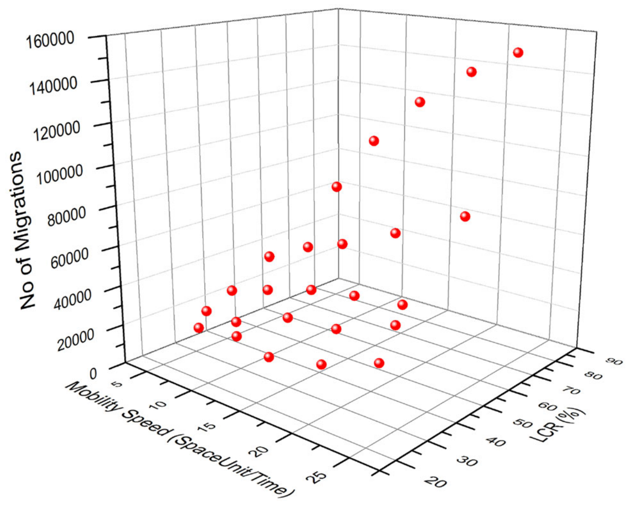

Fig. 4 Obtained LCR versus speed and number of migrations

SEs on 8 LPs, the reported LCR value is 24%. For moderate speed values, the obtained LCR value rises up to 81%. For higher mobility speed, a good clustering is obtained, however, leading to the escalation in the number of migrations.

5.2 LCR gain—comparison of MAHA and EHA

To obtain an insight into the performance of the proposed adaptive algorithm, with respect to the LCR obtained, the simulation is executed multiple times, by employing different number of LPs (i.e., 2, 4, 8, 16 and 32) during each simulation run. The partitioning of the simulation model into more and more LPs results in smaller clusters, which ultimately lead to the decrease in overall local communication. In this simulation, performance is investigated with respect to the ΔLCR gain and MR value when the number of LPs is in 2–32 range. Here, the ΔLCR is defined as the difference between the average value of local communication ratio when the migration status is on and off written as:

$$\Delta LCR = LCR_{MigrationON} - LCR_{MigrationOFF} \tag{15}$$

The positive value of ΔLCR means that the employed clustering technique is able to cluster the interacting SEs in a better way, whereas a negative or zero means that the approach is unable to cluster the SEs merely added the processing overhead. All the parameters of this simulation experiment are same as that of simulation except MR and the mobility speed that is set to 11 spaceunits per timestep. Figure 5 presents the results of gain in local communication (ΔLCR) when the simulation is executed multiple times for different number of LPs (2, 4, 8, 16 and 32) in each execution. The results in Fig. 5 demonstrate that for a moderate number of LPs, the proposed algorithm is able to

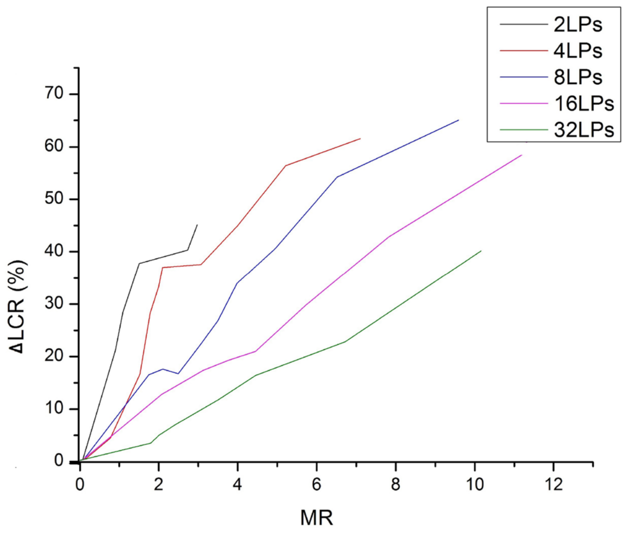

Fig. 5 Effect of the number of LPs on ΔLCR (LCR gain)

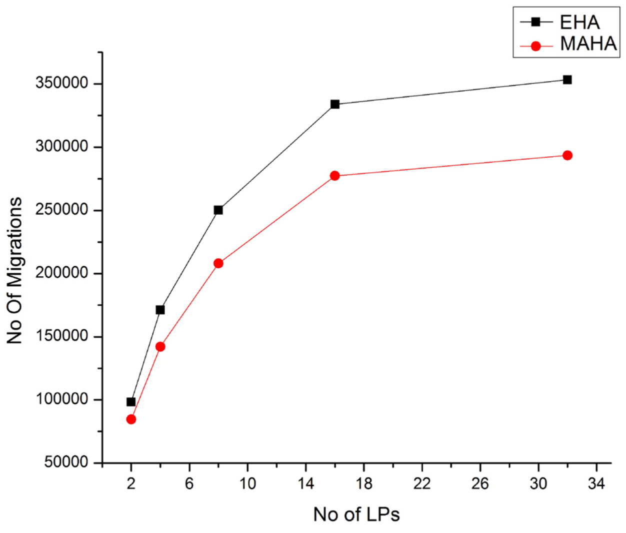

Fig. 6 Migration comparison of proposed MAHA versus EHA

achieve large LCR (61 to 65%) gain. When the simulation is divided into smaller partitions, it will be more difficult to produce effective clustering among the interacting SEs, even though there is still some LCR gain.

The results in Fig. 6 presents the details regarding number of migrations when the simulation model is partitioned into a different number of LPs during each simulation run. The simulation is executed multiple times with parameters stated in the initial discussion of Sect. 5.1. The obtained results demonstrate 15 to 22 % decrease in the number of migrations, while the attained LCR is same for both the (MAHA and EHA) approaches.

5.3 Simulation speedup comparison

The last simulation experiment is executed to measure the performance of the proposed adaptive algorithm's attained gain in terms of simulation execution speedup on the A-SIM-Cumulus Cloud for both parallel and distributed setup. The obtained results are then compared to that of the EHA. The simulations run over parallel (multi-core [41]) and distributed setup and the obtained execution time (simulation time (WCT)) with and without the migration is compared. For the parallel execution of the simulation, the A-SIM-Cumulus multi-core VM instance is configured with ARTIS/GAIA middleware. The number of cores allocated to VM is eight and the installed physical memory reserved for VM is eight GB. The simulations are executed multiple times and with different interaction probability (π) that is $\pi = 0.2, 0.5$, and 0.8, while keeping the value of MF to 1.1. This means that if the value of π is set to 0.5, then 50% of the SEs sends interaction (messages) during each timestep. The migration state size is set to 100, 45000, and 80000 bytes. The value 100 is default migration size for the simulations whereas 45000 and 80000 are obtained by padding extra bytes. In addition, the size of the communication message during each interaction is kept 1, 100, and 1024 bytes. The obtained results are average of several independent runs.

In Table 1, the obtained results pertaining to the execution time over parallel setup are reported. The simulation runs multiple times, with different configurations (interaction size and migration size) for three different dissemination probability values (Low, Moderate, and High) represented by $\pi = 0.2, 0.5$, and 0.8 respectively. In all cases, the simulation runs with migration status set to OFF and recorded the time (SimT) taken to complete the simulation execution. Afterwards, the simulation is executed, using the proposed MAHA algorithm and EHA. To evaluate the performance of the proposed approach, ΔSimT is used to show the obtained speedup. The ΔSimT is defined as the difference between the time taken to complete the simulation with and without migration. It is noted that ΔSimT is used to show the attained speedup for both the approaches (i.e., MAHA and EHA). For all the reported results, the migration approach is able to obtain a performance gain (2.73% for EHA and 3.64% for MAHA) as highlighted in Table 1. The worst case results are obtained when the migration size is increased up to 80000 bytes and the interaction size is 1 byte. Even though the migration mechanism is able to save costly remote communication for low Local Communication Cost (LCC). However, this results in an increase of cost paid for the migration and thus the gain in performance is narrow. On the other hand, the best results obtained are 20.39% for EHA and 28% for

Table 1 Parallel setup with migration OFF/ON. Different migration and interaction sizes. Different dissemination probability (π)

Input Parameters			$\pi = 0.2$				$\pi = 0.5$				$\pi = 0.8$			
			EHA		MAHA		EHA		MAHA		EHA		MAHA	
Migration status	Migration size	Interaction size	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT
OFF	–	1	110	–	110	–	245	–	245	–	324	–	324	–
ON	100	1	102.5	6.82	102	7.27	223	8.98	217.5	11.22	281.5	13.12	263	18.83
ON	45,000	1	105	4.55	104.5	5.00	234	4.49	221.8	9.47	287	11.42	271	16.36
ON	80,000	1	107	2.73	106	3.64	236	3.67	226	7.76	291	10.19	280	13.68
OFF	–	200	118	–	118	–	279	–	279	–	395	–	395	–
ON	100	200	108	8.47	104.5	11.44	256	8.24	242	13.26	339.5	14.05	326	17.47
ON	45,000	200	109.5	7.20	108	8.45	262	6.09	249.1	10.72	341	13.67	332	15.95
ON	80,000	200	110.4	6.44	109.5	7.20	263.2	5.66	251	10.04	342.5	13.29	340	13.92
OFF	–	1024	147	–	147	–	365	–	365	–	461	–	461	–
ON	100	1024	122	17.01	117.5	20.07	295	19.18	273.5	25.07	367	20.39	332.5	27.87
ON	80,000	1024	126.2	14.29	121	17.69	302	17.26	280.2	23.23	369.4	19.87	344	25.38

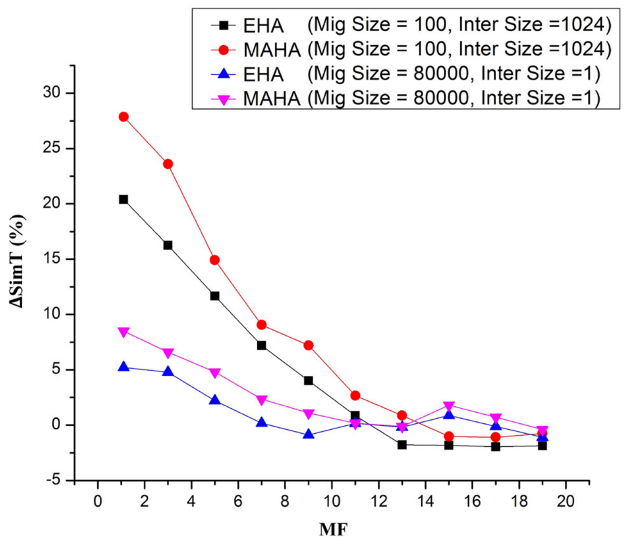

Fig. 7 ΔSimT for parallel setup when the MF value is in the range 1–19

MAHA when the interaction size is large (i.e., 1024 bytes) and the migration size is small (i.e., 100 bytes). It is worth noting that the best results are obtained when the MF is decreased to 1, thus leading to an increase in the number of migrations. The increase in the interaction dissemination probability also has a greater impact on the increase in the total interaction among SEs in the simulation. To obtain a detailed impact of MF on the performance of the proposed algorithm, the worst case and best case results are further investigated. For all the values of MF, the gain and loss are reported in Fig. 7. When the value of MF is set to 1.1, means that the migration mechanism is be able to obtain a large number of migrations. However, there will be no migration as the MF value is increased to 19. The results in Fig. 7 reveal that for MF value in the range of [1–10], obtain a performance gain. However, the increase in MF beyond 11 leads to an increase in the execution time.

The simulation is also executed on the distributed setup (A-SIM-Cumulus Cloud). The detailed specification for Cloud instances used in the simulation is given in Table 2.

The parameters chosen for parallel simulation are also used for distributed simulation setup. The obtained simulation results are reported in Table 3. All the configuration in terms of migration size and interaction size is tested for different dissemination probability (i.e., low, moderate, and high). Initially, the simulation runs without migration option and the simulation time (SimT) is measured for different runs. The obtained results are then stated as ΔSimT as shown in Table 3. ΔSimT is the difference between the time required to execute the simulation, with and without migration. In order to find the best configuration, the simulation is executed with different MF value in the range from 1 to 19. The best and worst case results for both MAHA and EHA are represented as italic face in Table 3. The best results obtained on EHA is 64% and on MAHA bumped up to 80% when the interaction size is 1024 and the migration size is 100 bytes. The gain is positive for the simulation where the value of MF is less than or equal to 10 which tends to decrease when MF >10. The best case and worst case speedup (ΔSimT) for the distributed simulation configuration are plotted in Fig. 8. The worst case results obtained is − 25% for EHA when the migration size is set to 80000 and small interaction size (i.e., 1 byte) and − 19% for MAHA. The results obtained

Table 2 Platform specification

Machine	CPU	Memory	No of cores	Operating system
VM1	Intel Core i3 1.70 GHz	4GB	4	Ubuntu 12.04.5 LTS
VM2	Intel Core i7 3.40 GHz	4GB	4	Ubuntu 12.04.5 LTS
VM3	Intel Core i7 3.40 GHz	8GB	8	Ubuntu 12.04.5 LTS
VM4	Intel Core i5 3.20 GHz	8GB	8	Ubuntu 12.04.5 LTS

Table 3 Distributed setup with migration OFF/ON. Different migration and interaction sizes. Different dissemination probability (π)

Input parameters			$\pi = 0.2$				$\pi = 0.5$				$\pi = 0.8$			
			EHA		MAHA		EHA		MAHA		EHA		MAHA	
Migration status	Migration size	Interaction size	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT
OFF	–	1	801	–	801	–	2178	–	2178	–	3178	–	3178	–
ON	1000	1	745	6.99	730	8.86	1965	9.78	1899	12.81	2789	12.24	2610	17.87
ON	45,000	1	819	− 2.25	779	2.75	1998	8.26	1926	11.57	2876	9.5	2705	14.88
ON	80,000	1	825	− 3.00	804	− 0.37	2023	7.12	1982	9.00	2901	8.72	2789	12.24
OFF	0	200	1159	–	1159	–	2350	–	2350	–	3590	–	3590	–
ON	1000	200	1046	9.75	989	14.67	2110	10.21	2056	12.51	3019	15.91	2676	25.46
ON	45,000	200	1098	5.26	1026	11.48	2170	11.76	2110	10.21	3112	13.31	2870	20.06
ON	80,000	200	1121	3.28	1067	7.94	2208	6.04	2156	8.26	3270	8.91	2915	18.80
OFF	0	1024	2890	–	2890	–	7550	–	7550	–	9866	–	9866	–
ON	1000	1024	1199	58.51	989	65.94	2879	61.84	1767	76.60	3566	63.86	1984	79.89
ON	45,000	1024	1265	56.23	1140	60.94	3150	58.28	1925	74.50	3965	59.81	2450	75.17
ON	80,000	1024	1304	54.88	1189	58.86	3357	55.30	2377	68.52	4488	54.51	2689	72.74

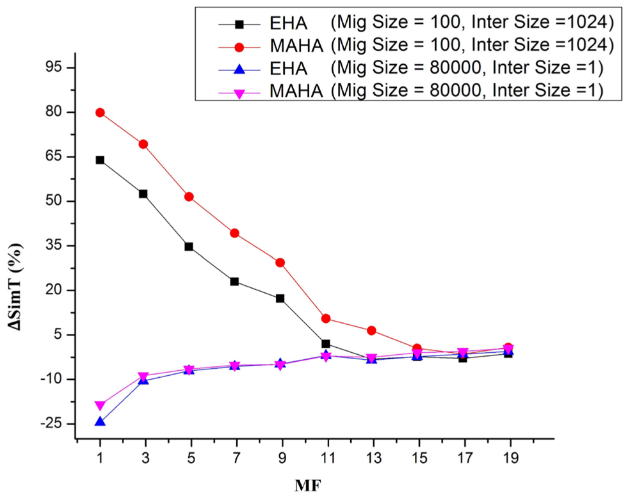

Fig. 8 ΔSimT for distributed setup when the MF value is in the range 1–19

from both parallel and distributed are not comparable due to the difference in hardware specification. From the given results it can be said that the lookahead delay has a clear impact on the time required to complete the simulation execution. Lookahead delay is the time required for synchronization among the LPs when an SE is to be migrated from one LP to another LP. With the increase in lookahead delay, the total cost of migration is also increased and thus increase in the number of migrations will adversely affect the execution time, however LCR will be improved. This will lead to a decrease in overall gain.

Table 4 Speed obtained when migration is applied for distributed, EHA, and MAHA approach

Number of nodes	10,000			20,000		
Number of LPs	2	4	8	2	4	8
Input parameters						
Distributed approach [28]	1.2	1.27	1.3	1.24	1.29	1.35
EHA [5]	1.4	1.51	1.65	1.42	1.55	1.68
MAHA1.2	1.56	1.72	1.78	1.59	1.76	1.82

Table 4 presents the results pertaining to the speedup obtained for three different approaches when migration is applied. The simulation is executed for 10,000 and 20,000 nodes with three different LPs combination that is 2, 4, and 8 LPs during each simulation run. For distributed approach, the maximum attained speedup is 1.3 and 1.35 for 10,000 and 20,000 nodes respectively when simulation is distributed and executed over 8 LPs. The highest speedup is observed for the proposed MAHA approach for both the simulation execution that is 1.78 and 1.82 respectively for 8 LPs with 10,000 and 20,000 nodes in the simulation. The obtained results reveal that the proposed approach is scalable and better in terms of the speedup achieved as compared to the distributed as well as EHA approach.

In summary, the proposed migration algorithm is able to achieve speed up for all the tested configuration on the parallel setup. Even though the magnitude of gain is limited in some configurations but relevant. For distributed simulation in most of the tested configurations, the results are much better in terms of performance gain, however, in some cases, the results are dropped down to negative.

6 Conclusion and future work

This research work opens new directions in the area of large-scale PADS. In this work, we have proposed an adaptive migration-based heuristic algorithm namely MAHA for large-scale network simulations over the Cloud. To support large-scale simulations in the Cloud, ARTIS/GAIA framework is integrated with SIM-Cumulus Cloud. The obtained results (for both multi-core and distributed simulations) demonstrate that the proposed migration algorithm significantly reduces the number of migrations and achieves better LCR. In addition, the proposed approach is able to achieve speedup in terms of execution time on both the multi-core and distributed architectures using A-SIM-Cumulus instances. In the case of distributed PADS, all the LPs follow heterogeneous architecture in terms of hardware resources and simulation communication load. An LP may become overloaded due to the computation required for handling huge communication (i.e., local as well as remote), which may result in LP crash. This will lead to the failure of overall simulation execution. To resolve this issue, in our future work, an LP migration approach will be proposed that could detect the LP load at run time and migrate the corresponding LP accordingly.

References

1. Benkhelifa, E., Welsh, T., Tawalbeh, L., Jararweh, Y., Basalamah, A.: Energy optimisation for mobile device power consumption: a survey and a unified view of modelling for a comprehensive network simulation. Mob. Netw. Appl. **21**(4), 575–588 (2016)
2. Bahwaireth, K., Benkhelifa, E., Jararweh, Y., Tawalbeh, M.A., et al.: Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications. EURASIP J. Inf. Secur. **2016**(1), 15 (2016)
3. Fujimoto, R.: Parallel and distributed simulation, in: Proceedings of the 2015 Winter Simulation Conference, pp. 45–59 (2015)
4. Mubarak, M., Carothers, C.D., Ross, R.B., Carns, P.: Enabling parallel simulation of large-scale hpc network systems. IEEE Trans. Parallel Distrib. Syst. **28**, 87–100 (2015)
5. Angelo, G.D.: The simulation model partitioning problem: An adaptive solution based on self-clustering. Simul. Model. Pract. Theory **70**, 1–20 (2017)
6. Fujimoto, R.M.: Parallel and Distributed Simulation Systems, vol. 300. Wiley, New York (2000)
7. Rhodes, J.D., Upshaw, C.R., Harris, C.B., Meehan, C.M., Walling, D.A., Navrátil, P.A., Beck, A.L., Nagasawa, K., Fares, R.L., Cole, W.J., et al.: Experimental and data collection methods for a large-scale smart grid deployment: methods and first results. Energy **65**, 462–471 (2014)
8. Zehe, D., Knoll, A., Cai, W., Aydt, H.: Semsim cloud service: large-scale urban systems simulation in the Cloud. Simul. Model. Practice Theory **58**, 157–171 (2015)
9. Angelo, G.D., Marzolla, M.: New trends in parallel and distributed simulation: from many-cores to cloud computing. Simul. Model. Practice Theory **49**, 320–335 (2014)
10. Rittinghouse, J.W., Ransome, J.F.: Cloud Computing: Implementation, Management, and Security. CRC Press, Boca Raton (2016)
11. Zeng, X., Bagrodia, R., Gerla, M.: Glomosim: a library for parallel simulation of large-scale wireless networks, in: ACM SIGSIM Simulation Digest, Vol. 28, IEEE Computer Society, pp. 154–161 (1998)
12. Angelo, G.D.: Parallel and distributed simulation from many cores to the public Cloud, in: High Performance Computing and Simulation (HPCS), 2011 International Conference on, IEEE, pp. 14–23 (2011)
13. D'Angelo, G.: Artis: design and implementation of an adaptive middleware for parallel and distributed simulation, in Technical Report, (2005)
14. Ibrahim, M., Iqbal, M.A., Aleem, M., Islam, M.A.: Sim-cumulus: An academic Cloud for the provisioning of network-simulation-as-a-service (nsaas), IEEE Access (2018)
15. Boukerche, A., Fabbri, A.: Partitioning parallel simulation of wireless networks, in: Proceedings of the 32nd conference on Winter simulation, Society for Computer Simulation International, pp. 1449–1457 (2000)
16. Szymanski, B.K., Saifee, A., Sastry, A., Liu, Y., Madnani, K.: Genesis: a system for large-scale parallel network simulation, in: Proceedings of the sixteenth workshop on Parallel and distributed simulation, IEEE Computer Society, pp. 89–96 (2002)
17. Group, H.W., et al.: Ieee standard for modeling and simulation (m&s) high level architecture (hla)-framework and rules, IEEE Standard 1516–2000 (2000)
18. Raczy, C., Tan, G., Yu, J.: A sort-based ddm matching algorithm for hla. ACM Trans. Model. Comput. Simul. **15**(1), 14–38 (2005)
19. Kumova, B.İ.: Dynamically adaptive partition-based data distribution management, in: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society, pp. 292–300 (2005)
20. Cai, W., Turner, S.J., Gan, B.P.: Hierarchical federations: an architecture for information hiding, in: Parallel and Distributed Simulation, 2001. Proceedings. 15th Workshop on, IEEE, pp. 67–74 (2001)
21. Boukerche, A., Tropper, C.: A static partitioning and mapping algorithm for conservative parallel simulations, in: ACM SIGSIM Simulation Digest, Vol. 24, ACM, pp. 164–172 (1994)
22. Boukerche, A., Das, S.K.: Dynamic load balancing strategies for conservative parallel simulations, in: Parallel and Distributed

Simulation, 1997., Proceedings., 11th Workshop on, IEEE, pp. 20–28 (1997)
23. Yocum, K., Eade, E., Degesys, J., Becker, D., Chase, J., Vahdat, A.: Toward scaling network emulation using topology partitioning, in: Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium on, IEEE, pp. 242–245 (2003)
24. Vigueras, G., Lozano, M., Orduña, J.M., Grimaldo, F.: A comparative study of partitioning methods for crowd simulations. Appl. Soft Comput. **10**(1), 225–235 (2010)
25. Angelo, G.D., Ferretti, S., Ghini, V.: Distributed hybrid simulation of the internet of things and smart territories, Concurrency and Computation: Practice and Experience
26. Angelo, G.D., Ferretti, S., Ghini, V.: Modeling the internet of things: a simulation perspective, in: High Performance Computing & Simulation (HPCS), 2017 International Conference on, IEEE, pp. 18–27 (2017)
27. Ferretti, S., D'Angelo, G., Ghini, V., Marzolla, M.: The quest for scalability and accuracy: Multi-level simulation of the internet of things, arXiv preprint arXiv:1710.02282
28. Logan, B., Theodoropoulos, G.: The distributed simulation of multiagent systems. Proc. IEEE **89**(2), 174–185 (2001)
29. Peschlow, P., Honecker, T., Martini, P.: A flexible dynamic partitioning algorithm for optimistic distributed simulation, in: Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society, pp. 219–228 (2007)
30. Angelo, G.D., Bracuto, M.: Distributed simulation of large-scale and detailed models. Int. J. Simul. Process Model. **5**(2), 120–131 (2009)
31. Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L.: Performance analysis of a parallel and distributed simulation framework for large scale wireless systems, in: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, ACM, pp. 52–61 (2004)
32. Angelo, G.D., Ferretti, S.: Simulation of scale-free networks, in: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 20 (2009)
33. Serrano-Iglesias, S., Gómez-Sánchez, E., Bote-Lorenzo, M.L., Asensio-Pérez, J.I., Rodríguez-Cayetano, M.: A self-scalable distributed network simulation environment based on cloud computing. Clust. Comput. **21**(4), 1899–1915 (2018)
34. D'Angelo, G., Ferretti, S., Marzolla, M., Armaroli, L.: Fault-tolerant adaptive parallel and distributed simulation, in: Distributed Simulation and Real Time Applications (DS-RT), 2016 IEEE/ACM 20th International Symposium on, IEEE, pp. 37–44 (2016)
35. I. Eucalyptus Systems, "Eucalyptus community cloud," http://open.eucalyptus.com/try/community-cloud [online] Accessed on
36. Zhou, A.C., He, B., Ibrahim, S.: "A taxonomy and survey of scientific computing in the cloud," Big Data: Principles and Paradigms, Morgan Kaufmann, eScience and Big Data Workflows in Clouds
37. Wainer, G.A., Mosterman, P.J.: Discrete-Event Modeling and Simulation: Theory and Applications. CRC Press, Boca Raton (2016)
38. Qu, Y., Zhou, X.: Large-scale dynamic transportation network simulation: a space-time-event parallel computing approach. Transp. Res. C **75**, 1–16 (2017)
39. Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M.: Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomput. **68**, 1–48 (2014)
40. Musolesi, M., Mascolo, C.: Mobility models for systems evaluation, in: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 43–62. Springer, New York (2009)
41. Yang, C., Chi, P., Song, X., Lin, T.Y., Li, B.H., Chai, X.: An efficient approach to collaborative simulation of variable structure systems on multi-core machines. Clust. Comput. **19**(1), 29–46 (2016)

Muhammad Ibrahim completed his Ph.D. in Computer Science from Capital University of Science and Technology (CUST), Islamabad in 2019. Currently, he is working as Lecturer at Virtual University of Pakistan. His area of research include Large-scale Network Simulation and Modeling, VM Migration, and Task Scheduling in Cloud Computing.

Muhammad Azhar Iqbal completed his Ph.D. in Communication and Information Systems from Huazhong University of Science and Technology, Wuhan, China in 2012. Currently, he is working as Lecturer in SWJTU-Leeds Joint School at Southwest Jiaotong University, Chengdu, China. His research interests include Internet of Things, Routing in Wireless Ad hoc Networks, and Large-scale Simulation Modeling and Analysis of computer networks in Cloud.

Muhammad Aleem received the Ph.D. degree in computer science from the Leopold-Franzens-University, Innsbruck, Austria in 2012. His research interests include parallel and distributed computing comprise programming environments, multi-/many-core computing, performance analysis, cloud computing, and big-data processing. He is currently working as an Associate Professor at National University of Computer and Emerging Sciences (FAST-NUCES), Islamabad, Pakistan.

Muhammad Arshad Islam completed his Doctorate from University of Konstanz, Germany in 2011. His dissertation is related to routing issues in opportunistic network. His current research interests are related to MANETs, DTNs, social-aware routing and Graph Algorithms. He is currently working as an Associate Professor National University of Computer and Emerging Sciences (FAST-NUCES), Islamabad, Pakistan.

Nguyen-Son Vo received the Ph.D. degree in communication and information systems from Huazhong University of Science and Technology, China, in 2012. He is with the Institute of Fundamental and Applied Sciences, Duy Tan University, Ho Chi Minh City, Vietnam. His research interests focus on self-powered multimedia wireless communications, quality of experience provision in wireless networks for smart cities, IoT to disaster and environment management. He received the Best Paper Award at the IEEE Global Communications Conference 2016 and the prestigious Newton Prize 2017. He has been serving as an Associate Editor of IEEE Communications Letters, 2019; Guest Editor of Elsevier Physical Communication, Special Issue on “Mission Critical Communications and Networking for Disaster Management”, 2019; Guest Editor of IET Communications, Special Issue on “Recent Advances on 5G Communications”, 2018; and Guest Editor of ACM/Springer Mobile Networks and Applications, Special Issues on “Wireless Communications and Networks for 5G and Beyond”, 2018 and “Wireless Communications and Networks for Smart Cities”, 2017.