Decreasing Communication Latency through Dynamic Measurement, Analysis, and Partitioning for Distributed Virtual Simulations

Robson E. De Grande, Azzedine Boukerche and Hussam Ramadan

Abstract—Communication in distributed simulations is highly relevant due to latencies cumulatively introduced in the execution time of such simulations; these latencies are caused by the transfer of data throughout networks. High Level Architecture (HLA) is a well known standard used to organize and keep the consistency of distributed virtual simulations. Nevertheless, although the HLA framework presents solutions to decrease communication overhead for simulations, it does not solve the communication issues that are pertinent to the network distances among federates and the Run-Time Infrastructure (RTI) in large-scale environments. Due to the relevance of load balancing for distributed simulations, many solutions have been proposed, but such approaches just consider communication load partially in their schemes or are limited. In order to minimize the communication overhead of HLA distributed simulations, a hierarchical dynamic balancing scheme composed of three phases was designed; however, even though the balancing scheme reacts properly to run-time load changes, its load analysis does detect all the imbalances after measuring simulations’ communication load. Thus, an extension is proposed to improve the detection of communication imbalances, as well as the redistribution of federates, so the balancing scheme can better react to communication load measurements gathered from each federate. In order to observe the benefits of the proposed communication balancing scheme and its extension, extensive large-scale experiments were realized. The results show that the scheme reduces considerably the amount of communication overhead, and the extension enables the detection of communication imbalances by modifying the decision making technique in the balancing system.

Index Terms—Distributed Simulations, HLA, Performance, Load Analysis

I. INTRODUCTION

Distributed virtual simulations are directly dependent on an environment’s shared resources, which require some kind of coordination to partition their load; however, the performance of such simulations are highly impacted by the management of dependencies between virtual simulations’ distributed elements. These dependencies are represented by interactions that can cause cumulative delays and overheads, which indirectly reflect on overall simulation execution time. Such overheads are even more evident when distributed virtual simulations are deployed on large-scale environments, using the HLA-based standard. In this case, the interaction latencies are increased by communication distances. These distances are produced by the deployment of simulation elements and by simulation intra-interaction behaviour, which can change dynamically and unpredictably. Based on deterministic factors, such virtual simulations can be statically partitioned and deployed on available resources; however, this partitioning is unable to reach a reasonable deployment of simulation elements when unpredictable load changes exist. Therefore, in order to keep or improve the execution performance of these distributed simulations, a dynamic scheme is devised to constantly measure resources’ computation load and simulations’ communication rate, analyze the execution conditions, and perform the needed load modifications.

The High Level Architecture (HLA) framework [1] provides a standard for designing and coordinating parallel and distributed virtual simulations. According to the HLA standard, distributed simulations are composed of federates, which are interactive, independent elements delimited by the HLA specification and organized by the Run Time Infrastructure (RTI) management services. Such services are responsible for maintaining the simulation consistency by controlling time, interaction, and data constraints. These management services also provide mechanisms to minimize communication overhead in HLA simulations, such as Data Distribution Management (DDM). DDM avoids the consumption of network resources by restricting the transmission of data that is relevant among the simulation federates. However, even with only existing relevant interactions, the DDM service cannot decrease communication delays that originate from the network distances between federates and the communication latencies inherent to these distances.

Moreover, since the distribution of load and the dependencies considerably affect the performance of virtual simulations, many schemes have been proposed to solve the performance issues regarding computation load and communication imbalances. Generally, these approaches mostly observe the load imbalances caused by computation factors; nevertheless, some approaches consider communication through aspects that involve simulation intra-dependencies. These communication aspects are usu-
ally either employed as an additional parameter to aid the decision-making of computation load redistribution in some schemes or are restricted to some re-partitioning policies or applications. The presence of such restrictions impedes the utilization of these communication balancing schemes for HLA RTI based simulations, in which the RTI acts as the central, global simulation coordinator.

In order to react to dynamic, irregular distribution of load, any balancing system needs to be able to detect load discrepancies. This detection is enabled by constantly measuring and analyzing a distributed simulation and the resources on which simulation elements are running. Even though this approach reorganizes HLA simulations specific only to communication aspects, computational load is considered in the redistribution scheme. This metric is employed to avoid load imbalances and to provide awareness of resources processing status, which is collected through resource management systems. A resource management system provides tools for administrating shared resources, which include the needed monitoring of resources. The Grid system, a resource management system, is amply used to manage distributed resources and applications, aiming to enable a coordinated, flexible, secure resource sharing system [2]. Therefore, due to its characteristics that embrace reliability, security, and flexibility, Globus Toolkit [3], a de facto Grid system middle-ware, is accessed by the proposed balancing scheme to gather data regarding monitoring measurements of distributed virtual simulations.

Due to the need for a balancing mechanism that considers the aspects of HLA-based virtual simulations, a scheme has been proposed in [4]. This scheme constantly measures the load of shared resources and a distributed simulation’s communication rates, analyzes them, and performs the proper modifications. The proposal of this scheme aims at a more comprehensive communication balancing system that can minimize the network latencies by employing a more general redistribution analysis. Such a balancing system considers the proximity of resources to federates’ communication destinations. The system is designed under a hierarchical structure that decreases the overhead caused by measuring and analyzing simulation entities, facilitating the management of simulations. The balancing scheme divides the balancing process in three inter-dependent sequential phases: monitoring of resources and federates, re-allocation of resources for federates, and execution of migration moves. However, even though the proposed scheme is effective in decreasing communication delays in simulations, it shows a limited detection of communicative federates. The detection technique employed in this scheme does not adapt to abrupt and severe communication changes. Due to the magnitude of the balanced simulations, the collected load data sample might hide communication rates that represent an overload of federates. Since such overload is not detected, the balancing system cannot modify the arrangement of federates to decrease communication delays, and different analysis techniques are required to better detect communicative federates. Therefore, extensions for the detection phase of the balancing scheme are proposed in this paper, so the balancing scheme can react properly and achieve a more precise deployment of federates.

The remainder of this paper is organized as follows. In Section 2, the related work is introduced, and the challenging issues are presented. In Section 3, the proposed scheme, as well as its structure, architecture, and balancing phases are described. In Section 4, two extensions for the balancing scheme are delineated. In Section 5, experiments are explained and their results are discussed. Finally, the conclusion briefly outlines and describes the directions for future work.

II. Related Work

Due to the complexity of providing a load balancing solution for optimistic and conservative distributed simulations, many balancing approaches have been designed to considerably improve performance. All the proposed schemes encompass computation load and simulation intra-dependencies, which reflect on communication load. Measuring the computation load and taking the proper actions benefit simulations considerably. However, such a practice is ineffective when simulations present communication imbalances. Consequently, measuring and analyzing simulations’ internal interactions is highly relevant for execution performance. Following this proposition, several approaches consider the simulation look-ahead and the communication dependencies between simulation federates. The analysis of these metrics is performed statically, before the simulation is run, or dynamically, during run-time. However, these schemes are restricted to specific simulation applications and cannot be applied to any HLA RTI based simulations.

Inevitably, many balancing systems mainly consider either computation load of shared resources or simulation speed of a distributed simulation to re-deploy distributed simulations. In a resource centred paradigm, schemes basically focus on measuring the direct result from imbalances, which corresponds to the CPU utilization of virtual simulation federates or their execution throughput [5] [6] [7] [8] [9] [10] [11]. In this approach, the main criteria is to equalize the consumption of resources, so a simulation can obtain the maximum benefit from them. In simulation centered paradigms, schemes measure the simulation speed of each federate and compare it to the average simulation speed of a whole simulation [12] [13] [14] [15]. Such an approach focuses on the process that is slowing down the simulation; consequently, this relieves the resource where the process is running and can benefit the entire simulation. However, even though the measuring of computation load can evidence the need for changes, it might not detect imbalances caused by simulation dependencies or by delays originated from communication latencies.

Some other balancing mechanisms assign certain relevance to interactions between simulation entities and consequently attempt to eliminate or decrease the delays caused by internal dependencies in distributed simulations. Such simulation internal dependencies are evidenced by the
differences in simulation look-ahead and communication load. The look-ahead is a monitoring metric that can efficiently show the pace of a simulation entity and hence enable the discovery of a slow simulation entity that is causing delays in other simulation elements [16] [17]. The dependencies, which are reflected on the communication delays, are employed to determine a simulation distribution that minimizes such latencies. In this scope, some techniques analyze such dependencies statically [18] [19] while others measure, detect, and modify simulations dynamically [20] [17] [21] [22] [23] [24] [25] [26]. Among these dynamic balancing systems, some just add the communication information to the computation balancing; others mix the computation load and communication dependencies or measure the communication load to redistribute the simulation. However, these techniques are simulation-dependent, limit the simulation parallelism, or disregard the network topology of shared resources.

Therefore, all the existent approaches for re-partitioning the communication load of distributed simulations cannot be totally applied to HLA RTI based simulations. The limitations of detecting and re-deploying simulation entities restrict the balancing schemes to specific simulation applications or disregard the network topology of the resources that are used to run the simulations. The previous approaches are successful in reducing the communication overhead and the simulation delays, but they are limited to peer-to-peer re-deployment of simulation load, ignoring the proximity of shared resources for the migration of load. A proximity analysis allows the system not only to benefit from the creation of more possibilities of minimizing communication latencies but also to enable communication balancing for simulations based on centralized RTIs. A dynamic balancing scheme [4] that observes the proximity of resources has been designed, and it effectively decreases the communication latencies in HLA based simulations. This balancing scheme uses proximity analyses to determine the re-arrangement of federates according to their distance in the communication topology. However, such a scheme presents some limitations in detecting discrepancies of communication load. Through the scheme’s detection mechanism, certain highly communicative federates are not considered for redistribution due to particularities of data samples collected from a large-scale virtual simulation. Moreover, the detection of such communicative federates is vital for enabling a highly efficient re-deployment of simulation federates. Thus, extensions are proposed to improve the detection mechanism of this balancing scheme, more effectively determine communication imbalances, and redistribute load accordingly.

III. PROPOSED BALANCING SCHEME

The main goal of the proposed balancing scheme in [4] is to employ the distribution of resources and to take advantage of their proximity in order to provide an application-independent solution that decreases or minimizes the communication latencies for large-scale HLA RTI based distributed simulations. Due to dynamic load and communication changes that might occur during a simulation execution, the balancing scheme needs to constantly monitor a simulation and its environment. This periodic measuring provides means to responsively redistribute the simulation federates according to balancing requirements in order to minimize the communication latency. The scheme consequently is divided in three sequential balancing phases that periodically measure the needed parameters and react to imbalances. Moreover, the proposed balancing system is structured hierarchically and employs a low latency migration procedure. The hierarchical structure decreases the overhead and latencies caused by the balancing system when it measures and redistributes simulations’ load.

As shown in Figure 1, the architecture of the balancing scheme basically consists of a Group Manager (GM) and several Local Management Agents (LMA). All these elements are placed on shared resources by following a hierarchical order, and they are all implemented in Java, aiming at a cross-platform design and integration with the RTI. Even though the scheme performs redistribution of simulations considering only communication aspects between federates, it is vital that the balancing system accesses third-party mechanisms to obtain computation load of shared resources. This information is required in the decision making part of the redistribution algorithm to establish a re-configuration of simulation elements, so that simulations do not undergo overload caused by computation load imbalances.

The GM corresponds to the main element in the balancing scheme; it administrates a set of LMAs that are assigned to it and organizes the balancing phases: monitoring, redistribution, and migration. GMs are organized in a hierarchical structure in which each GM is responsible for managing sub-GMs, which are GMs in a lower hierarchical level, used for reporting to an upper GM about its collected data sample. For the monitoring, a GM requests information from its set of LMAs and sub-GMs and accesses a third-party tool. Each LMA, as well as each sub-GM, forwards information regarding the communication behaviour of simulation federates. The third-party tool is accessed in order to collect information regarding the load of the shared resources that a GM is responsible for managing. The GM filters and analyzes the collected information in order to identify communication discrepancies. After the analysis is performed, a redistribution is generated according to the load of the shared resources. At the end of the process, a GM emits migration calls, which are forwarded to the respective LMA.

A LMA is responsible mainly for acting as an interface between a GM and some simulation federates. A LMA is placed in each shared resource that is eligible for receiving a federate. Each LMA contains a set of Communication Load Monitors and Migration Managers. A Communication Load Monitor acts together with a federate and constantly logs the federate’s communication. The Communication Load Monitor answers periodical requests from its respective LMA with logs’ data. A Migration Manager, likewise a Communication Load Monitor, is respon-
sible for managing the migration procedure of each federate. Triggered by a migration call forwarded by its LMA, the Migration Manager is required to support a migration. This support essentially prevents simulation inconsistencies, which are caused by lost messages or the arrival of non-ordered simulation events. In a migration procedure, a Migration Manager launches a Migration Manager remotely, suspends a federate’s execution, restores its state, and coordinates the required data exchanges. The Migration Manager also activates a Migration Proxy to assist the federate migration when the destination resource of a migration call is unreachable by the Manager, so a peer-to-peer data exchange cannot be realized. The Migration Proxy acts as an intermediate migration element that has the role of forwarding data to a migrating federate.

In order to be suited for large-scale environments, a hierarchical architecture is introduced to organize all the balancing system, decrease the overhead generated by the balancing scheme, and facilitate the management of HLA virtual simulations. The hierarchy is organized in several layers, and each GM is responsible for coordinating a set of LMAs and a set of sub-GMs, as stated previously. Each LMA is an end-point or a leaf in the structure and basically corresponds to a resource in the distributed system. GMs that control just LMAs are in the bottom of the hierarchy and are managed by other upper layer GMs. The role of this bottom GM is to forward monitoring information to an upper GM or migration calls to a LMA. In the intermediary layers of the structure, the GMs also forward information and migration calls, but they additionally filter and merge information. The GM located at the top of the hierarchy is responsible for indirectly or directly managing all other GMs; this GM performs global data gathering and balancing analyses, redistributes federates accordingly, and emits migration calls.

A. Monitoring Phase

The distributed simulation is balanced periodically, and the monitoring phase is activated at the beginning of each balancing cycle. This phase is essential for enabling responsiveness to the dynamic balancing system, and based on it, the other subsequent phases are triggered. The monitoring is responsible for measuring the communication rate of simulations and computation load of the environment; it also employs detection techniques to identify communication dependencies that delay simulations. The computation load information is collected through the access to a third-party tool, the Monitoring and Discovery Service from Grid Services [27]. This information is used in the decision-making part of the re-distribution phase. The communication rate is gathered from each Communication Load Monitor, which keeps the interactions of each federate in a communication table. This table is located in the same resource as the federates, and it logs every federate’s interaction. A federate’s interaction is comprised of a destination address and the number of messages sent and received (as well as their size). More particularly, inasmuch as the simulations are implemented based on an HLA RTI centralized architecture, every federate communicates strictly and directly with the RTI; consequently, the destination address in the log tables are all the same. All this communication information is sent to a GM every monitoring interval to perform filtering and selection.

In every collected data sample, selection is applied to identify conclusive aspects that are used as decision factors. Such aspects correspond to discrepancies in communication rates of federates, exhibiting that the re-arrangement of federates on resources might decrease communication latency. The selection simply consists of performing comparisons between each federate’s communication rate and their overall communication rate average. Through this selection technique, the most communicative federates of a simulation in a given moment are differentiated. The communication rate average is retrieved through the calculation of an arithmetic mean; also, a threshold is utilized to determine the communicative federates more precisely. Initially, the value obtained from the calculation of the standard deviation of the communication rate average is
employed as the selection threshold, which corresponds to a superior boundary when added to the average. The communication rate is an application-dependent metric, which shows a behaviour that changes according to each simulation implementation, so standard deviation is used in order to produce a parameter that is totally based on the collected system’s communication rates. After the selection is realized, the re-distribution phase is invoked in order to re-arrange the federates as required.

B. Redistribution Phase

After an ordered list of communicative federates is selected, a re-partitioning is performed to search for the most appropriate destination resources for such federates. These federates are re-allocated by evaluating them according to the re-distribution procedure described in Algorithm 1. This re-partitioning of federates is performed with the objective of precisely determining migration moves to destination resources that can benefit simulation performance by decreasing the communication latencies; consequently, a classification of resources is realized according to their network topology to match with the communication rate of each federate candidate. This classification mainly employs the distances between resources and a specific destination, which is the RTI in this particular application case. As a result, highly communicative federates are elected to be moved to resources that are located closer to the RTI, so the communication latencies are reduced, decreasing delays in simulations.

Following the ordered list of federates and according to the location of each federate, the redistribution algorithm searches for the proper destination resource for each communicative federate. A data structure is used by the algorithm to store all the positions of the shared resources. This structure facilitates the search for a resource that is close to the target destination of a determined federate. By observing this structure and looking just to the communication scope, the most appropriate resource to transfer a federate to is the one that the federate communicates with the most. This resource is the optimal destination for the transfer of this federate because the federate’s communication latencies are eliminated or reduced considerably. This approach is extensively employed by the previous communication balancing solutions, which group together the same resource simulation entities that have intense communication among them, avoiding networking delays. However, this reduces the parallelism of distributed simulations. This approach is also considerably limited and disregards the possibility of overloading resources, evidencing the impossibility of applying it to simulations based on centralized HLA RTIs. Therefore, a different technique is used to enable the migration of federates to resources close to a RTI and not to the same resource where the RTI is running. In this case, the path distances structure is essential for determining resources nearby the RTI.

The Path Distances structure organizes all the shared resources in distance rings according the analysis the communication topology. Each ring contains a group of resources that have the same distance to the RTI. The distance is defined as the sum of the number of network hops between two resources, which are weighed by their network capacities. In each distance ring in the structure, the resources are organized according to their computation load, so the balancing scheme is able to identify a resource that does not harm the distribution of computation load, avoiding the generation of load imbalances. Since the network topology is characteristically static and the destination is the same for any simulation federate, the rings of the path distances structure are generated only once: in the initialization of the balancing scheme. However, the ordering of resources in each ring is determined by the computation load of resources in a given moment.

Regarding the communication aspects, only the nominal network capacity in each hop and number of hops are used in the Path Distances structure to define distances, considering that the hop with the smallest nominal transmission capacity is used to qualify a distance. Resources are classified according to this static network configuration to define the destination for communicative simulation entities. These metrics represent a near static state of the networking resources, but the addition of dynamic factors would increase the complexity of the balancing system. Although the use of delay variations to fit federates’ communication needs to network capacities presents a more realistic solution, its feasibility can lead to considerable balancing delays. By observing the network capacities and attempting to fit federates according to the devices’ available bandwidth promotes the problem to a higher level of complexity; an example of this is the solution of the bin packing problem, a NP-hard problem. For instance, many factors, such as communication distance, individual and global communication rate, communication flow, and others, need to be measured because they directly or indirectly influence delays. A deep search and analysis caused

<table>
<thead>
<tr>
<th>Algorithm 1 Communication Redistribution Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Require:</strong> current.loads, spec.loads, federates.loads, path.distances</td>
</tr>
<tr>
<td>order(federates.loads)</td>
</tr>
<tr>
<td>cmean = calc_mean(federates.loads)</td>
</tr>
<tr>
<td>cstd = calc_STD(federates.loads, cmean)</td>
</tr>
<tr>
<td>comm.federates = find_comm(federates.loads, mean, std)</td>
</tr>
<tr>
<td>for all federate IN comm.federates do</td>
</tr>
<tr>
<td>while 'resource_found' and path.distances(next)</td>
</tr>
<tr>
<td>ring = get_closest_ring(path.distances, RTI)</td>
</tr>
<tr>
<td>resource = least_load_resource(ring)</td>
</tr>
<tr>
<td>if 'overloaded(resource.load)' then</td>
</tr>
<tr>
<td>migration_move.add(federate, resource)</td>
</tr>
<tr>
<td>resource_found = TRUE</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end while</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>return migration_moves</td>
</tr>
</tbody>
</table>
by this complexity increase results in higher balancing latencies that can lower the responsiveness for the redistribution mechanism. Therefore, in a matter of simplification that minimizes the balancing overhead, the delays are not considered in the detection and redistribution phases.

The resource with the least load and which is located at the distance ring closest to the RTI is selected as the destination resource candidate to receive the migrating federate. After the selection, the resource candidate needs to be analyzed in order not to become overloaded because of the federate migration. In this case, an analysis evaluates the load of the candidate by considering the average load of the rest of the shared resources. The resource candidate’s computation load is compared with the overall computation load of the resources. The difference between the load values needs to obey the policies delimited for computation load distribution [28] in order not to harm the computation load balance, as delimited in Algorithm 2. In the case that a resource candidate is not able to receive a federate for migration, the next candidate is searched in the distance rings; thus, the next ring in the structure is selected, consequently identifying the resource with the least load. The same comparisons are consecutively applied to determine a migration move for the communicative federate. The procedure of searching for a resource candidate in the distance rings continues while the destination resource is not found or the communication latency of a ring is not greater than the latency of the resource where the federate is running. The stop condition of the algorithm establishes that any communication improvement cannot be reached since the situation of the federate cannot be improved in a given load configuration of the distributed system.

Algorithm 2 Evaluation Algorithm

Require: src_rsc, dst_rsc
if dst_rsc < min then
    if number fed(src_rsc) > 1 then
        create migration move(src_rsc, dst_rsc)
    else if number fed(src_rsc) = 1 & src_rsc > (min $\phi$) then
        create migration move(src_rsc, dst_rsc)
    end if
else if (dst_rsc - src_rsc) < (min $\phi$) then
    if number fed(src_rsc) > 1 then
        create migration move(src_rsc, dst_rsc)
    else if number fed(src_rsc) = 1 & (dst_rsc - src_rsc) > (min $\phi$) then
        create migration move(src_rsc, dst_rsc)
    end if
end if
return migration move

C. Migration Phase

Migration is the final step of the balancing cycle in the scheme, and after the redistribution defines the federate moves that are necessary to be realized, the migration calls are emitted to their respective Migration Managers. The migration procedure can introduce substantial overhead to simulation execution time; consequently, the responsiveness of the balancing system is directly related to the migration latency. The reduction of the migration latency to a minimal time enables the system to produce a better performance improvement. Basically, the migration procedure consists of suspending the execution of a federate in a resource and restoring it at the chosen destination; however, more complexities exist in this procedure because simulation inconsistencies might be caused by the migration due to the time and data coordination relevance for an HLA simulation. The federate migration procedure is required to transfer the initialization files of a federate to the destination resource, suspend the federate’s execution, retrieve its execution status and the incoming messages, transfer its state and messages to the destination, and restore the federate execution at the destination. The data transfers represent the largest latencies in all the migration procedure, mainly considering it for large-scale environments. Thus, in order to minimize the migration latency, a two-phase migration mechanism is adopted in the balancing system. This technique is similar to the techniques proposed by Boukerche and De Grande [29] and Zengxiang et. al. [30]; also, it requires minimal external tools, avoids unnecessary communication and computation overhead, and does not introduce global synchronization in the simulation.

The first phase of the migration procedure is responsible for transmitting the static initialization files, which are required for starting up the migrating federate at the destination resource. After these files are transmitted, the Migration Manager can be configured properly to perform the next steps in the migration process. These files are transferred using a third-party tool, GridFTP [3]. The migrating federate and the respective Migration Manager are launched remotely through the Web Service Grid Resource Allocation and Management [3]. Access to third-party tools introduces substantial overhead in the migration process, but overhead is not incorporated into the migration latency because the federate is not suspended while the Migration Manager is not completely initialized at the remote destination.

The second phase in the migration procedure corresponds basically to suspending a federate and restoring its execution at the destination resource. Initially, the Migration Manager triggers the exchange of the communication channels in the RTI, so the migrating federate starts receiving messages instead of the federate at the source resource. After the exchange is successfully accomplished, the Migration Manager sends a call for the suspension of the federate’s execution. Then, the federate saves its execution state and the messages that it received and did not process. The Migration Manager sends both state data and messages to the remote resource. The execution state is comprised of the dynamic information (variables) that represents the current execution status of a federate and its Local Run-time Controller’s state. When the state and the messages are received, the Migration Manager at the
remote location passes them to the migrating federate. Finally, the federate restores its state and takes over its execution.

IV. Extension of the Scheme

The proposed dynamic balancing scheme employs a non-complex analysis technique, acting like a greedy algorithm. Even though it is a simple approach, the used analysis provides a fast solution that is close to an optimal simulation redistribution. With this approach, the scheme obtains a rearrangement of load in a short time. The quick balancing answer minimizes the consumption of resources, without introducing balancing overhead to the system that is managed; this technique also increases the responsiveness of the balancing system, enabling a more frequent detection and redistribution of simulation load. Furthermore, regarding only the observed communication aspects, the redistribution algorithm achieves an optimal solution, in which the network distances are minimized according to interaction rates of communicative federates. Nonetheless, considering computation factors in the redistribution algorithm, the balancing problem is reduced to a sub-optimal solution. Because these factors are considered to avoid computation overload of the shared resources, they interfere with the communication objectives and can impede a communicative federate’s movement to a closer location.

Moreover, the communication rate is a factor that is attached to each specific simulation implementation and consequently is obtained through calculations regarding only the gathered data sample. Consequently, the analysis of this metric evidences certain communication particularities, which represent the considerable communication overhead. However, the analysis may not identify some particularities. In such an analysis, the calculation of a common overall high average of federate communication rates can hide the overhead caused by some communicative federates. As a result, the balancing scheme might determine that the system is balanced, but the distributed system does not reach reasonable performance improvement even if migrations are realized to reduce delays.

Therefore, the detection of imbalances and the redistribution of simulation load in the balancing scheme are based on a mechanism that does not fully identify communicative federates. The mechanism employs standard deviation for calculating the threshold used to detect federates with high communication rates. The standard deviation tends to follow the load behaviour of an analyzed data sample, so the use of standard deviation to identify communicative federates impedes the detection of some communication imbalances or hides them. Such behaviour is observed when the collected data sample is composed of extreme values. The extreme communication rates lead to a steep increase or decrease in the calculation of the mean, augmenting the standard deviation value. The increase of such a value, caused by the high variation of communication rate, makes the detection and redistribution algorithms focus on the extreme cases and not identify other non-extreme high communication rates. Consequently, the use of just standard deviation to select and determine redistribution can hamper the dynamic balancing for such simulations.

Even though extreme values of communication rates are rare for some data samples, they can be produced frequently by distributed elements in certain applications, evidencing a common data pattern. For instance, in the experiments of the previous work [4], the simulation scenario was composed of several federates that intensively published objects’ updates while a few federates were subscribed to these updates and received an overwhelming number of messages. These recipient federates represented stations that constantly monitored the positioning of objects and their respective firefighting situation in a given moment. In this scenario case, message recipients presented the largest communication rates and became the bottleneck in the simulation. These communication rates led the standard deviation to assume a large interval, which excluded other communicative federates from the redistribution analysis, as shown in Figure 2. The figure describes the previous results obtained using the previous redistribution approach [4] in such a simulation scenario. In such results, the number of migrations is constantly limited to three in every simulation case. These migrations explicitly depict how the detection of imbalances was misled by a few communicative federates, which received all the object updates during the simulations and kept an extremely high communication rate. Thus, the standard deviation cannot be applied to every case; other approaches are required to determine a general, better suited reallocation of distributed elements to improve the performance of a distributed simulation.

As mentioned previously, two parts of the balancing algorithm are influenced by these extreme communication rate values and need to be modified/extended: detection of imbalances and redistribution of federates. The detection of imbalances shows which parts of an application/simulation are causing imbalances and the indication of load changes that are essential to achieve a better execution performance. As a consequence, the redistribution determines the required modifications with the information that is provided by the detection phase. If the detection is not performed properly, the rearrangement of distributed elements is limited. A better approach to detect the imbalances is needed, so extreme cases are considered but do not influence the analysis of the rest of the data sample. For the extension, two approaches are proposed: one applies iterative filtering in the detection/redistribution process and the other employs limiting conditions different from the standard deviation.

A. Federate Filtering

In this extension, a filtering of federates is performed to specify the analysis of gathered data samples. The communicative federates that were moved to a more favourable location continue to be monitored by the balancing system. Since such federates might present a dynamic communication behaviour, they need to be observed constantly; thus, the balancing scheme needs to analyze all the federates
and all the resources in order to improve the distribution of simulation parts. However, the consideration of this global view in the analysis can mislead the identification of communication imbalances in certain particular cases, as stated previously. In order to avoid this situation, recursive filtering is applied on the data sample. This filtering increases the threshold’s precision that determines the communicative federates.

The detection threshold’s precision is influenced by the distances of nodes that are considered in the calculations. A further analysis that performs filtering and/or selection is employed to allow the detection of imbalances belonging to non-extreme samples. Currently in the balancing scheme, the distances of resources to certain specific locations are employed to determine redistribution of load. In the filtering, the distances are used to incrementally search the federates that cannot benefit from the balancing algorithm. Therefore, the recursive filtering determines which federates are relevant for the detection of imbalances and the reallocation of resources. This cumulative filtering is added to Algorithm 1 and is performed before the search for closer resource destinations starts.

In the detection algorithm (together with the load redistribution), the distance of each federate’s resource is used to remove a federate from the data sample. The federates that are considered overloaded through an initial communication load analysis are selected for a distance analysis. This distance analysis observes the proximity of such federates to their communication destination. Because these federates are already in the closest available topological position, no migration move can be performed to decrease their communication overhead. The presence of such federates can hamper the detection of other communicative federates, which can have their communication latencies decreased through federate migrations. Consequently, as an extension, if a communicative federate is running on a resource that is located in the closest position to the destination, the federate is discarded from the analysis. With this procedure, other federates which have a moderate but relevant communication rate are considered in this extended detection process. Thus, more interactive federates can be identified in the detection phase.

The process of identifying the communicative federates that are closer to their communication destination is performed repetitively until the system reaches a defined condition. This process influences the analysis of the communication balance of the system, improving the reduction of communication overhead produced by the distributed elements. Generally, whenever the search algorithm identifies a communicative federate to be excluded from the list of federates, a recalculation of mean and its standard deviation is performed on the data sample. With this recalculation, new overloaded federates are identified and analyzed in order to refine the search or to produce migration moves to improve the simulation performance. However, this process needs to have a well-defined condition to reach a final deployment of federates. Such a condition is determined by a threshold that is obtained by summing the average and the standard deviation calculated after the first federate filtering. Therefore, through this approach, the issues caused by the extreme communication loads are avoided; moreover, this extension of the algorithm produces a more extensive analysis than when employing just the standard deviation to classify federates as overloaded.

B. Extensive Federate Analysis

A second extension for the detection/re-distribution approach is designed to improve the re-distribution of federates by increasing the balancing convergence speed. In this extension, the algorithm checks all the federates by maintaining a condition which enforces certain limitations and which is thereby more comprehensive than the one stated previously. This condition is composed of a federate analysis and a resource analysis. The first one provides the number of federates that require migration to produce a better simulation performance. The second analysis determines the availability of resources to receive such federates. In this analysis, computation load analysis is employed as a determinant factor to conduct this extension. As a result, the balancing system redistributes federates while imbalances exist or migrations are still possible.

Observing the inability to detect communicative federates caused by the use of standard deviation in the analysis, the second extension excludes the utilization of such a value. The limiting condition is determined by the relation between the initial average and the median of the data sample. The incorporation of the median into the analysis provides a general description of the communication rate
distribution in the collected data sample. This new analysis shows a general view of the amount of communicative federates present in a simulation. The analysis disregards the federate’s communication rate, preventing miscalculations. Therefore, the limiting condition for the communication balancing analysis is denoted by the smallest amount between the two statistical values, \( \min(\text{average}, \text{median}) \).

Through this approach, the issues caused by the extreme communication loads are totally avoided.

With this technique, communication balancing evaluates a more ample set of federates, which also includes those with a relatively moderate communication rate. Consequently, this extended algorithm produces a more extensive analysis than when employing just the standard deviation to classify federates as overloaded. The use of a low-value threshold induces a selection of a larger number of federate candidates to be checked as communicative. Generally, the threshold determines the analysis of at least half of the data sample in order to determine migration moves properly. Through this extensive evaluation, the detection step certifies that communicative federates in certain cases are not excluded from the redistribution procedure. As a consequence of a more extensive analysis, the approach introduces additional overhead to the balance processing. However, this additional overhead, which is larger than the previous extension approach and the original detection algorithm, is still noticeably less than the overhead caused by the evaluation of an entire distributed simulation. In this extension, the number of evaluations is limited to half of the simulation federates if resources are available. In a large-scale overloaded system, this processing reduction becomes crucial to achieve reasonable measuring and analysis efficiency.

Allying both federate and resources analysis parameters in the balancing scheme, this extension generalizes the procedure of detection and redistribution by re-allocating federates to available non-overloaded resources. Basically, the extension of the approach attempts to assign all the available resources to simulation federates in order to decrease the distance between them and their communication destination. These two parameters, the number of communicative federates and the number of available resources, condition the balancing. The detection/redistribution algorithm finishes when one of these parameters reaches zero. In this case, either all the resources in a better position are used for redistributing the load, or all active federates are re-allocated to a closer position. Therefore, this extension makes the procedure of detection and redistribution more general through a cyclic reallocation of shared resources to simulation federates.

Like the previous communication redistribution algorithms, the list of federate candidates is analyzed sequentially, generating migration moves for the most communicative federates first. Communicative federates are assigned to a closer resource and eliminated from the list. This process is realized while closer resources are available. The algorithm accesses the Path Distance structure to retrieve information from the available resources and updates this structure when a migration assignment is determined. For every federate, the algorithm needs to perform a search in a sequential pattern in the structure’s rings, always starting the search from the beginning of the Path Distance that contains the list of resources. In this particular case, the beginning of this structure is the ring closest to the RTI.

The resource search in this extension is formed in this manner because federates are organized by their communication rate and are definitely assigned to a destination resource after analyses of computation load are performed on resources in the rings. Since the computation load of resources is used in the decision-making part of the redistribution and is not employed to organize the list of federates, resources can be assigned to a larger number of communicative federates. Thus, with this approach, even if there is no considerable communicative federates in the data sample, federates are still assigned to resources. The technique attempts to move any federate that presents a higher communication rate in order to decrease the communication delays and to balance distributed simulation.

V. Experiments

In order to identify the efficiency improvement of the proposed extension to the dynamic balancing scheme, three experiments have been realized. These experiments are similar to those accomplished in [4]. The experiments showed that the extension could decrease the communication latencies, improving the simulation’s performance even more. Such experiments were supported by a large-scale distributed environment composed of two computing clusters connected by a fast-Ethernet network link. One cluster, a Dell cluster, consists of 24 nodes, while the other one, an IBM cluster, is composed of 32 nodes. In the Dell cluster, each node comprised a Quadcore 2.40GHz Intel(R) Xeon(R) CPU and 8 gigabytes of DIMM DDR RAM memory. All the Dell nodes were inter-connected through a Myrinet optical network that allowed data transmission up to 2 gigabits per second. In the IBM cluster, each node consisted of a Core 2 Duo 3.4 GHz Intel(R) Xeon(R) CPU and 2 gigabytes of DIMM DDR RAM. A gigabit Ethernet network connected the IBM cluster’s nodes. The fast-Ethernet link connecting both clusters cannot present all the aspects originated from the Internet; nevertheless, the link is introduced to produce a communication bottleneck for simulations, avoiding unexpected external influences and evidencing the need for balancing. Moreover, Linux operating systems have been installed in both clusters. In order to support all the experimental environments, the HLA platform with an RTI version 1.3 and the Globus Toolkit 4.2.1 have been used in the simulations.

In the experimental virtual simulations, the simulation federates were evenly placed on the 55 shared nodes, and the HLA RTI executive was deployed on a dedicated node. For the balancing system, the LMAs were placed on all of the cluster nodes except the ones that were designated to the RTI and the GMs. There were two GMs, and each one was placed in one node of each cluster, considering that one
GM was the root. Furthermore, in the simulation scenario, federates simulated an emergency preparedness scenario of a firefighting situation. In the simulation, federates coordinated the actions to update the information of objects that represented firefighters, fire focuses, and buildings in a two-dimensional routing space in time-stepped virtual simulations. Such federates updated the information of their objects that were published and subscribed to interest spaces. The simulations were composed of 500 federates that managed 1 to 1000 objects in 100 time steps. In order to produce controlled, differentiated communication latencies in the simulations, certain federates performed the publication of special objects that generated a large communication overhead.

The first experiment measured the performance gain that the balancing system provided to a HLA virtual simulation. In this experiment, the dynamic balancing scheme that was proposed previously was compared with the two aforementioned extensions. As depicted in the graph in Figure 3, an increasing communication overhead was applied in a small set of simulation federates: a federate that publishes special object updates, and three other federates that subscribe to them. The increase of communication load reflected in a proportional delay the simulations’ execution time. A noticeable performance gain can be observed in the graph when comparing the base-line, which is not balanced, with the other balanced cases. When observing just the curves of the three balancing schemes, the normal balancing system shows a slightly smaller performance gain in Figure 3(a), but when comparing the number of migrations among the schemes, the extension that performs extensive federate analysis presents a high number of migrations. By crossing the information between the graphs in the figure, the results evidence that the extensions are not more efficient than the normal balancing scheme for simulations with a few communicative federates, though performing more modifications in the simulation distribution.

In the second experiment, the responsiveness of all the balancing schemes is observed and compared when an
increasing communication overhead is introduced to the overall distributed simulation. In the experiment, 100 special objects were assigned among 40 federates. The curves in the graph of Figure 4(a) show that all the schemes reduced the communication overhead considerably by detecting latencies and reacting properly to them. Nevertheless, the curve that corresponds to normal balancing presented a performance decrease when compared with the curves of the extensions. When observing the number of migrations in Figure 4(b), the performance difference originates from the decrease in the number of migrations that occurred simultaneously with the increase of execution time in Figure 4(a). Such behaviour occurred due to the inability of the normal balancing to react properly when extreme communication rates are present in the collected data sample. This limitation does not exist in both extensions because of their more detailed analysis and extra filtering.

Finally, the third experiment measured the capacity of the balancing systems to detect and react to dynamic changes of communication load. This experiment is similar to the previous one, but its communication load changed dynamically instead. The federates that were selected to present communication overhead changed their load randomly during the execution time of a simulation, producing an unpredictable behaviour. In this experiment, 1 to 60 federates in the simulations produced a random amount of object updates that ranged from 1 to 100. As shown in Figure 5(a), the balancing schemes presented a large performance gain when compared to the base-line, and all their curves showed similar performance improvement; however, by observing the curves in more detail, the normal balancing scheme shows a performance decrease between 30 and 50 federates when compared with only the extensions. Such behaviour is caused by the overall communication overhead in the system that was not detected by the normal balancing scheme, which employs standard deviation to detect imbalances. The same behaviour is observed in Figure 5(b) when the curve of the normal balancing scheme is compared with the curve of the balancing with extra resource filtering. This particular behaviour is caused by the communication load produced in the experiments. With load less than 30 federates, the communication rates were not as high as needed to mislead the normal balancing. Between 30 and 50 federates, the communication load was enough to produce a few extreme communication rates, which influenced the normal balancing but not the extensions. With a load of more than 50 federates, the larger number of communicative federates (less load variations in the data sample) caused the normal balancing to have a narrower standard deviation, inducing a better detection of communication imbalances. Thus, through this experiment, it was noticed that the extensions proved to be more effective than the old balancing scheme in some cases.

VI. Conclusion

In this paper, a dynamic balancing scheme for decreasing communication latencies from HLA RTI based virtual simulations was analyzed and compared with two proposed extensions. Based on a multi-layered hierarchical structure, the scheme is divided in three inter-dependent sequential phases; these detect communication imbalances, redistribute simulation federates according to communication latencies, and transfer federates between shared resources. These three phases basically consist of constantly measuring the communication rate of distributed virtual simulations and analyzing load distribution to determine corrections for imbalances. Moreover, Grid services are accessed by the balancing system in order to add reliability and security to the system when measuring the computation load of shared resources for redistribution decision-making and performing data transfers for migrations.

The experiments proved the efficiency of the balancing scheme in detecting and reacting properly to dynamic communication rate changes. Both extensions improved the balancing scheme, enabling it to more effectively detect communication load changes and to produce a rearrangement of federates that was able to decrease the simulation execution time. Even though the gains that the extensions provided were not substantially large when compared to the normal balancing scheme, the extensions...
were more efficient in balancing the simulations in specific situations. These specific situations caused the detection step of the normal balancing to fail. Moreover, a particular case was observed in the experiments when comparing results of both extensions: the extensive analysis showed the same/similar performance gain in all experiments, but the number of migrations that it produced to balance simulations was considerably higher than other schemes. This particular situation happened due to the trade-off between migration latency and responsiveness. The extension performed many migrations, but the migration latencies consumed the performance gains. Thus, as future work, aiming at exploring the trade-off between migration latencies and performance gain, further experimental analysis will be realized to produce a balancing scheme that can produce a better balance with less migrations. Another aspect to be explored is the addition of a computation load balancing system to the scheme, so the system will be able to react to dynamic simulation load changes. To provide a more realistic balancing scheme for large-scale simulations, communication delays will be analyzed by considering balancing overhead and performance gain.

ACKNOWLEDGEMENT

This work is partially supported by NSERC, the Canada Research Chair program, MRI/ORF research funds, the Ontario Distinguished Research Award, and the EAR Research Award.

REFERENCES