Predictive Dynamic Load Balancing for Large-Scale HLA-based Simulations †

Robson Eduardo De Grande and Azzedine Boukerche
PARADISE Research Laboratory - School of Information Technology and Engineering
University of Ottawa, Canada
Email: {rdgrande,boukerch}@site.uottawa.ca

***Abstract*—Due to the dependency on resources, HLA-based simulations can experience load imbalances and consequently loose execution performance. The High Level Architecture (HLA) was developed to facilitate the creation and control of distributed simulations, but such an architecture does not offer solutions for solving load imbalance issues. To provide mechanisms for preventing performance loss caused by load imbalances in distributed simulations, numerous balancing approaches have been developed. The majority of these mechanisms present limited awareness of environment characteristics. To cope with this problem, a distributed dynamic balancing scheme has been designed; however, its redistribution algorithm, as other developed balancing schemes, is limited and does not react properly in the presence of abrupt load changes due to be based on recent load status. Therefore, a predictive balancing scheme is proposed to provide a method to decrease the number of precipitated migration moves and to detect and prevent load imbalances based on load variation tendencies. In order to evaluate the proposed scheme, experiments have been conducted to analyze performance gain and efficiency.**

***Keywords*-Parallel Simulations; High Level Architecture; Load Balancing; Prediction; Performance.**

I. Introduction

With the growing need to design and implement complex systems, such as large-scale virtual environments, distributed simulations, largely HLA-based simulations, have been receiving increasing attention. Since these simulations are highly dependent on a distributed environment, they can undergo substantial performance loss due to heterogeneity of resources, presence of external background load, improper placement of simulation elements, or dynamic simulation load changes. Even though a static initial deployment of simulation entities (load) can avoid imbalances caused by the heterogeneous characteristics of resources, this deployment cannot prevent the unpredictable load changes. Therefore, a dynamic balancing system is needed to react to such load oscillations and improve performance; in this system, the awareness of load oscillations is improved through the utilization of prediction techniques that enable the prevention of some load variations.

† This work is partially supported by NSERC, the Canada Research Chair program, ORF funds, and EAR Research Award.

High Level Architecture (HLA) has been devised to provide a design framework and methods to facilitate the creation and coordination of distributed simulations. The general aim of this framework is the re-usability of simulation entities, called federates, and the interoperability of simulation parts. The framework is essentially structured on a set of design rules, object model templates, and an interface specification. Management services are provided through the Run-time Infrastructure (RTI) component to maintain the consistency of HLA-compliant simulations. Federates access the services to inter-communicate and progress with simulation processing. However, HLA only comprises the management of simulation aspects, and it is not able to prevent performance loss arising from load imbalances.

Aiming to prevent load imbalances, several load redistribution schemes have been developed. A great part of these schemes are either limited or do not completely consider the characteristics of distributed resources and applications. In order to overcome such drawbacks, a distributed balancing scheme has been designed [1]. This scheme presented a decentralized structure that enabled responsiveness to dynamic load changes from HLA-based simulations and external processes. However, the scheme is prone to acute load oscillations, leading the system to hastily and improperly produce unnecessary load modifications.

To provide means to decrease or prevent precipitated load rearrangement, a predictive balancing scheme is proposed. The scheme incorporates prediction metrics and techniques in the redistribution algorithm to enable the system to avoid reactions to abrupt load variations and to allow a preventive detection of some load imbalances. Based on a prediction model, the proposed system generates predictions, which require modifications of detection and load reallocation methods to analyze, identify, and react to significant and potential imbalances.

The remainder of this paper is organized as follows. In Section 2, the related work and challenging issues are described. In Section 3, the proposed predictive balancing scheme is detailed by showing its architecture and describing its algorithms. In Section 4, the experimental scenario and results are described and discussed. In Section 5, a brief conclusion is presented defining directions for future work.

II. Related Work

Load balancing is essential for increasing or keeping execution performance of distributed simulations. Because of the importance of balancing load, many schemes have been developed. Some designed schemes consider communication characteristics and dependencies between simulation entities to coordinate the reorganization of simulation elements. Other schemes observe the computational aspects of distributed simulations and shared resources to achieve a reasonable rearrangement of simulation load.

The analysis of simulation look-ahead and communication dependencies provide means to identify the simulation elements that increase the simulation execution time. The evaluation of look-ahead allows to detect the simulation entities that generate considerable delays for other simulation parts [2], [3]. The observation of communication rate enables the detection of delays and waiting times caused by the communication latencies. The analysis of communication dependencies can be performed statically [4], [5] or dynamically [6], [3], [7], [8], [9], [10], [11], [12], [13], [14]. This allows the network distances between communicative parts to decrease. Even though this type of balancing provides performance gain, it does not solve the performance issues caused by imbalanced load on shared resources.

Focused on rearranging distributed simulation elements according to their computational load, balancing schemes can be restricted to simulation-centred or resource-centred approaches. The simulation-centred solutions observe the load through simulation aspects, which allows to increase the simulation execution pace by improving the relative speed of each entity [15], [16], [17], [18]. The resource-centred solutions analyze the shared resources' load and aim to maximize the even consumption of shared resources [19], [20], [21], [22], [23], [24], [25], [10], [26]. All these developed solutions present limitations and do not fully cover the major balancing issues for large-scale distributed simulations, such as resource heterogeneity, causality inconsistencies, and external background load. In order to consider all these issues, a centralized balancing scheme [27] and a distributed scheme [1] have been designed.

The distributed balancing scheme, as well as the centralized one, showed that it is prone to abrupt load oscillations and is limited to the load status collected in the recent past. Due to high responsiveness of the balancing scheme, abrupt oscillations exercise substantial influence on it, generating unnecessary precipitated migration moves. The monitoring metric used in their algorithms restrict them to only correct imbalances after they have occurred and not before. Therefore, a predictive scheme is proposed to reduce the number of precipitated load transfers and improve the balancing responsiveness to load imbalances through preventive measures.

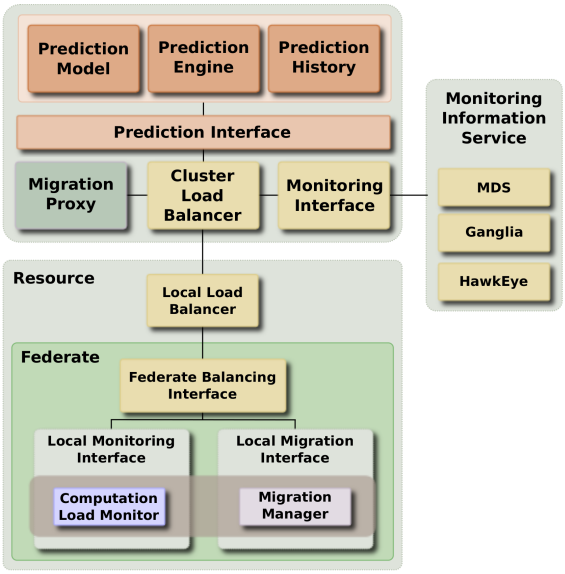


Figure 1: Dynamic Balancing Scheme's Architecture

III. Predictive Distributed Load Balancing Scheme

In order to build a predictive balancing scheme, prediction methods, functions, and components are incorporated into the distributed balancing scheme [1]. Moreover, the balancing process basically presents a similar order of tasks as the previously designed distributed scheme, which consists in monitoring resources, re-arranging simulation load, and migrating federates. The proposed architecture contains additional elements for calculating and providing predictions, and the protocols are modified to be able to manage policies based on load predictions.

As depicted in Figure 4, a Cluster Load Balancer (CLB) is responsible for managing a set of resources (cluster) and the federates placed on them. This management comprises of gathering load information from resources and federates, analyzing the collected data and data from neighbour CLBs, and triggering federate migrations. In order to coordinate the whole the balancing process, the CLB is connected with neighbour CLBs and accesses the Monitoring Interface, Local Load Balancers (LLBs), and the Prediction Interface.

The Monitoring Interface introduces transparency for the access to monitoring data from a Monitoring Information Service (MIS). Providing load information related to the cluster monitoring level, the MIS accesses Grid services for gathering application-independent load metrics. Grid is a resource management system widely used to coordinate the execution of distributed applications on shared resources [28]. The access to the resource load monitoring information is made through Grids Index Service, which obtains the data from Monitoring and Discovery Service.

An LLB acts as an interface for each resource and its

containing federates. Placed in every resource, LLB is focused on locally managing simulation federates by gathering monitoring data, applying data aggregation methods, and forwarding migration calls. The gathering of monitoring data consists of forwarding a call from a CLB through a LLB and its Federate Balancing Federates, which triggers each Local Monitoring Interface to retrieve information regarding a federate's CPU consumption. The dispatching of migration calls is triggered by a CLB call, which ends in the Migration Manager (MM).

The migration procedure is performed in two-phases, similarly to the federate migration described in [29] and [30]. The federate migration is divided in a transfer of static, initialization files and dynamic execution status data. Through Grid services, the first part of the transfer and initialization of the federate at the remote resource is performed. In the second part of the migration, a peer-to-peer transfer is realized between the MMs at the local resource and at the remote resource to enable a migrating federate to restore its execution state. A Migration Proxy is used for migrations between unreachable resources.

A Prediction Interface is introduced in the architecture to enable and facilitate the access to predictions of collected sample data in the balancing system. This interface allows transparent access to the Prediction Engine. The engine processes the incoming data based on a history and a prediction model. The prediction data history is stored in the Prediction History component. This element is dependent on the prediction model since different metrics need to be saved to generate a prediction. For the prediction model used in this work, smoothing and trend values are stored in a list. The list is updated according to the balancing system needs. The model is determined in the Prediction Model component and fed by the Prediction Engine with the information provided by the CLB and the Prediction History elements.

A. Load Redistribution Algorithm

The predictive balancing scheme presents a redistribution algorithm divided in three phases: monitoring of distributed load, rearrangement of simulation entities, and migration of federates. As delineated in Algorithm 1, in order to enable responsiveness to dynamic load changes, the balancing system is required to evaluate load distribution periodically. This consequently imposes the scheme to work in balancing cycles of Δ time units, which is limited by the data refresh rate of the monitoring tool.

The monitoring phase directly dictates the responsiveness of the balancing system through the timely provision of precise distributed load status to the balancing system. The precision and the interpretation of the collected data considerably influences the balancing performance, motivating new methods of evaluating the load distribution. In this case, prediction techniques are applied to the collected data,

Algorithm 1 Distributed Dynamic Load Balancing Algorithm

```
1: loop
2:    loads ⇐ query_MDS()
3:    loads ⇐ filter_MDS_data(loads)
4:    loads ⇐ normalize_loads(loads, benchmark)
5:    loads ⇐ prediction(loads, old_loads, mig_RSCs)
6:    old_loads ⇐ loads
7:    overload_cand ⇐ select_overload(loads)
8:    spec_loads ⇐ request_LLBs(overload_cand)
9:    mng_loads ⇐ filter(loads, spec_loads)
10:   mig_moves ⇐ local_bal(mng_loads, SP)
11:   mig_moves ⇐ local_bal(mng_loads, MP)
12:   mig_moves ⇐ local_bal(mng_loads, LP)
13:   send_migration_moves(mig_moves)
14:   if mig_moves = ∅ then
15:      data_neighbours ⇐ request_Neighbour_Load_Data()
16:   else
17:      if relFactor ≥ random_number(1, 100) then
18:         data_neighbours ⇐ request_Neighbour_Load_Data()
19:      else
20:         data_neighbours ⇐ ∅
21:      end if
22:   end if
23:   neighbours ⇐ identify_neighbour_Less_Load()
24:   while neigbours! = ∅ do
25:      overloaded_resources ⇐ select(firstNeighbour, SP)
26:      overloaded_resources ⇐ select(firstNeighbour, MP)
27:      overloaded_resources ⇐ select(firstNeighbour, LP)
28:      federates ⇐ select(spec_loads, overloaded)
29:      eliminate_first(neigbours)
30:   end while
31:   send_to_neighbour(overloaded_resources, federates)
32:   migration_moves ⇐ wait_for_migration_moves()
33:   send_migration_moves(migration_moves)
34:   wait( Δ )
35: end loop
```

requiring modifications on the balancing algorithm to bear this new evaluation of resources' and federates' load status.

After raw data concerning the resources' load status is retrieved, filtering is applied to eliminate unneeded or misleading values. The gathered data is then normalized through benchmarks to allow the appropriate comparison of resources' loads. Finally, predictions are calculated for each resource's load status value. The calculation involves the current load value, the previously smoothed load value, and the migration status of a resource. The previous smoothed load value is required by the prediction model, and the migration status is used to adjust the prediction parameters, improving the system's responsiveness. For each resource, short-term, medium-term, and long-term predictions are calculated. Detection is applied to determine overloaded and underloaded resources on the data sample.

Together with part of the monitoring, the redistribution phase defines a load rearrangement towards a decrease of imbalances. This phase also needs to be modified in order to accommodate the prediction metrics in its decision-making procedure. The analysis of future simulation load behaviour provides a predictive reaction to load oscillations and a decrease in the time gap between the instant in which data is gathered and the moment when the data is observed in the balancing system.

The redistribution phase initiates with the local balancing

Algorithm 2 Local Predictive Pair-Match Evaluation Algorithm

Require: $src\_rsc, dst\_rsc, min, \phi, \delta$
1: $min, \delta, \phi \Leftarrow adjust(src\_direction, dst\_direction, type)$
2: **if** $dst\_rsc < min$ **then**
3: **if** $number\_fed(src\_rsc) > 1$ **then**
4: $create\_migration\_move(src\_rsc, dst\_rsc)$
5: **else if** $number\_fed(src\_rsc) = 1$ & $src\_rsc > (min * \phi)$ **then**
6: $create\_migration\_move(src\_rsc, dst\_rsc)$
7: **end if**
8: **else if** $(dst\_rsc - src\_rsc) > (min * \delta)$ **then**
9: **if** $number\_fed(src\_rsc) > 1$ **then**
10: $create\_migration\_move(src\_rsc, dst\_rsc)$
11: **else if** $number\_fed(src\_rsc) = 1$ & $(dst\_rsc - src\_rsc) > (min * \phi)$ **then**
12: $create\_migration\_move(src\_rsc, dst\_rsc)$
13: **end if**
14: **end if**

Algorithm 3 Inter-Domain Remote Predictive Balancing Algorithm

Require: $external\_loads, internal\_loads$
1: **if** $external\_loads! = \emptyset$ AND $internal\_loads! = \emptyset$ **then**
2: $external\_loads \Leftarrow order(external\_loads, type)$
3: $internal\_loads \Leftarrow order(internal\_loads, type)$
4: **for all** $external\_loads$ **do**
5: **for all** $internal\_loads$ **do**
6: $src\_rsc \Leftarrow select(external\_loads)$
7: $dst\_rsc \Leftarrow select(internal\_loads)$
8: $mig\_moves \Leftarrow evaluate(src\_rsc, dst\_rsctype)$
9: $external\_loads \Leftarrow clean(external\_loads, mig\_moves)$
10: **end for**
11: **end for**
12: **end if**
13: **return** $external\_loads, mig\_moves$

Algorithm 4 Inter-Domain Predictive Pair-Match Evaluation Algorithm

Require: $int\_rsc, ext\_rsc, min, \delta, type$
1: $min, \delta \Leftarrow adjust\_Parameters(src\_direction, dst\_direction, type)$
2: **if** $int\_rsc < min$ **then**
3: $create\_migration\_move(int\_rsc, ext\_rsc)$
4: **else if** $(dst\_rsc - src\_rsc) > (min * \delta)$ **then**
5: $create\_migration\_move(int\_rsc, ext\_rsc)$
6: **end if**

calls for each type of prediction (SP, MP, and LP). The local balancing consists of matching the overloaded resources with the underloaded resources according to the required type of prediction. Three local redistribution processes are realized independently, giving greater importance to the prediction closer to the current time. At the beginning of the local redistribution, an ordering is performed according to the type of prediction that is defined. At the end of the procedure, all the migration moves are returned, and the respective resources are removed from the list for the next local balancing call. For either an overloaded or underloaded resource, a local pair-match analysis is performed, as described in Algorithm 2. In this analysis, if the difference of load between an overloaded and an underloaded resource exceeds a certain value and justifies a need for load transfer, a migration move is generated and returned.

The inter-domain balancing is activated based on the defined local migrations and on the success rate of the inter-domain migrations. Cluster load is then requested from the neighbour CLBs. This load consists in a set of load averages that represent a neighbour cluster's resources. In order to prepare these averages to be sent to the requesting CLB, the data sample (list of resources) is filtered by the resources contained in the Interquartile range of the resources' load list. The averages are returned to the caller CLB, which can proceed with its inter-domain balancing. With the averages from the neighbour CLBs, a selection is performed to identify imbalances between the domains. For each type of prediction value, resources with load larger than a threshold are selected as candidates for further, remote redistribution analysis.

Upon receiving resources selected for inter-domain, a neighbour CLB initiates its inter-domain redistribution analysis, which consists in evaluations for each type of prediction: SP, MP, and LP. The evaluations encompass sequential comparisons between the remote resources (overloaded) and the local resources (underloaded), as described in Algorithm 3. For each selected pair of resources, a comparison based on thresholds is performed, as described in Algorithm 4. If the load difference between the resources is large enough to justify an imbalance (threshold), a federate migration is generated. At the end of the selection, the migration calls are grouped and sent to the requester CLB, and resources involved with migrations are selected for correcting prediction calculations.

B. Forecasting Load Status

Independent of the prediction model employed in the balancing scheme, the load forecasting is performed on a short-term, medium-term, and long-term basis. As a result, different predictions enable the provision of cautious load changes through short-term predictions and preventive load redistribution through medium-term and long-term predictions. In the case of the balancing system, the period that defines the forecasting is based on balancing cycles; therefore, the short, medium, and long term cycles are defined as 1, 3, and 5 respectively.

In each balancing algorithm (local or inter-domain), adjustments are applied on the pair-match thresholds. These modifications on the parameters are dependent on the type of validation and uni-dimensional direction to which a resource's load is tending, as described in Algorithm 5. Predictions are not considered with the same weight in the comparisons; the thresholds for medium and long term predictions receive larger adjustments, which are proportional to their projection time.

The load tendency direction is also employed in the adjustment procedure, as detailed in Algorithm 5. This direction of load oscillation is calculated for each medium or long term prediction. The direction is simply obtained by calculating the difference between a projection and the short-

Algorithm 5 Predictive Adjustment

Require: $src\_direction, dst\_direction, type$
1: **if** $src\_direction \geq 0$ AND $dst\_direction < 0$ **then**
2: $\quad adjust(min, \delta, \phi, type, COND1)$
3: **else if** $src\_direction \geq 0$ AND $dst\_direction \geq 0$ **then**
4: $\quad adjust(min, \delta, \phi, type, COND2)$
5: **else if** $src\_direction < 0$ AND $dst\_direction < 0$ **then**
6: $\quad adjust(min, \delta, \phi, type, COND2)$
7: **else if** $src\_direction < 0$ AND $dst\_direction \geq 0$ **then**
8: $\quad adjust(min, \delta, \phi, type, COND3)$
9: **end if**
10: **return** min, δ, ϕ

term prediction. According to the algorithm, the directions of both source and destination resources are used in a simple mechanism to define a type of adjustment ($COND1$, $COND2$, and $COND3$). $COND1$ represents the situation in which thresholds are less tolerant to imbalances since both resources tend to increase the load gap between them. $COND2$ represents an intermediary situation in which the thresholds are slightly more tolerant to imbalances because one of the resources is stabilizing (inversion of load variation). Finally, $COND3$ represents the situation in which the thresholds are the most tolerant since both resources present inversion in their load tendency (stabilization).

Moreover, when load transfers are performed, load oscillations are produced, and these variations are not part of the simulation load behaviour. Thus, for the calculation of predictions, the prediction model's parameters are modified according to resources selected for migration. As described in the next subsection, the parameters receive values to give more emphasis to current gathered value than to the past load values, providing a more accurate adaptation of the load trend calculation.

C. Prediction Model

The balancing system gathers the computational load data in uniformly spaced time intervals. This list of values characterizes a natural temporal ordering, and it can be interpreted as a time series. Thus, techniques for processing and analyzing time series are applied to generate smoothing or to provide certain predictions. One such technique can be employed to obtain more steady current load values and to define determined future load variations.

Based on the Exponential Weighed Moving Average (EWMA) calculation, three exponential smoothing techniques [31] can be used to determine load predictions: single, double, and triple exponential smoothing. The first one offers a simple smoothing of data, totally based on the EWMA, without any extrapolation in the average calculation. On the other hand, triple exponential smoothing adds seasonality to the average computation, obeying fixed patterns and requiring before-hand configuration. These two methods are either limited or require previous knowledge about load oscillations; the double exponential smoothing is used as a prediction model since it employs the concept of trend in the smoothing calculation. Trends are highly relevant since they allow the identification of a tendency in the computational load and enable the production of forecasting.

The double exponential smoothing is applied on the gathered load data to produce certain forecasting and is represented by Formulas 1 and 2. The associated formulas define a smoothed value based on the current load $elem_i$ and the previously smoothed value sum_{i-1}. The trend is added as the factor obtained from the second formula. The trend enables the extrapolation of the average and provides means to determine a future tendency in observed data. In Formula 3, a forecasting is calculated based on the current smoothing value and its respective trend factor. The prediction of load variations is given by m, which represents the number of future time intervals for the predictive projection.

$$sum_i = \alpha \times elem_i + (1-\alpha) \times (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 \quad (1)$$

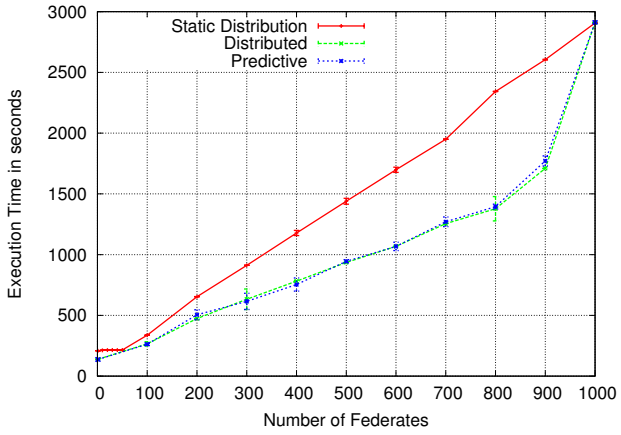
$$t_i = \gamma \times (sum_i - sum_{i-1}) + (1-\gamma) \times t_{i-1}, 0 \leq \gamma \leq 1 \quad (2)$$

$$f_{m+i} = sum_i + m \times t_i, m > 0 \quad (3)$$

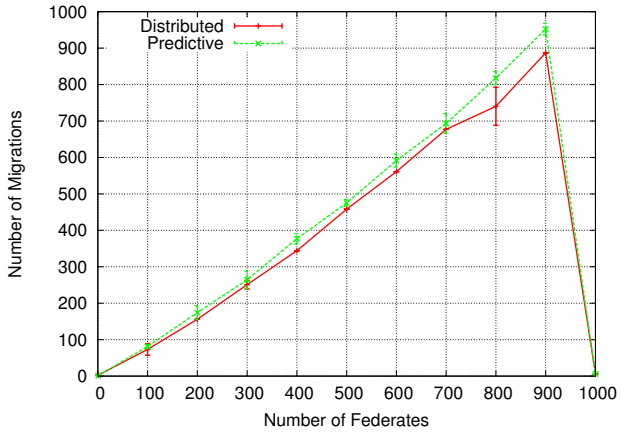
The smoothing method is used for any $i \geq 0$. Therefore, the initial smoothing is set as $sum_0 = elem_0$ because of the lack of previous smoothing values or the absence of at least one previous element to support the calculation. The calculation of the load trend requires at least two past smoothing values, so the initialization of the formula parameters comprehends assigning 0 to t_0 and performing $t_1 = elem_1 - elem_0$. Moreover, both smoothing formulas are delimited by two constants (α and γ); both smoothing and trending constants are set to work in conjunction.

IV. Experimental Results

To evaluate the proposed predictive balancing system, the system was compared with the distributed balancing system [1]. Thus, some experimental scenarios were delineated to realize the comparisons through efficiency and responsiveness analysis. The testbed of the experiments was an heterogeneous environment that consisted of simulations deployed on two (IBM and Dell) clusters of computing servers. The IBM cluster was composed of 32 computing servers interconnected by a gigabit Ethernet network; also, each server contained a Core 2 Duo 3.4 GHz Intel(R) Xeon(R) CPU and 2 gigabytes of DIMM DDR RAM. The Dell cluster presented 24 computing servers interconnected through a Myrinet optical network that allowed data transmission up to 2 gigabits per second; also, each server contained a Quadicore 2.40GHz Intel(R) Xeon(R) CPU and 8 gigabytes of DIMM DDR RAM memory. The two clusters were interconnected through their management nodes with a fast-ethernet link. Linux operating system was installed in

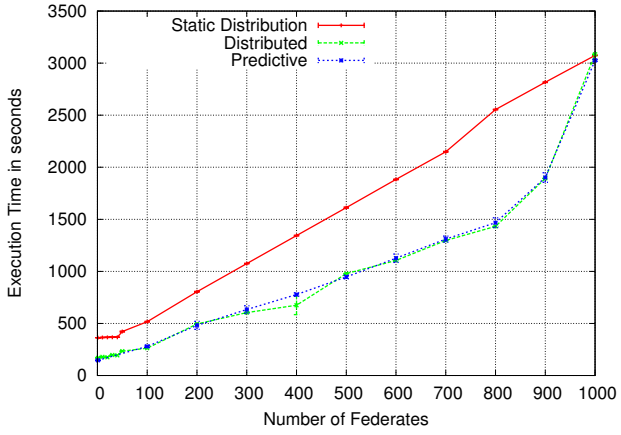


(a)

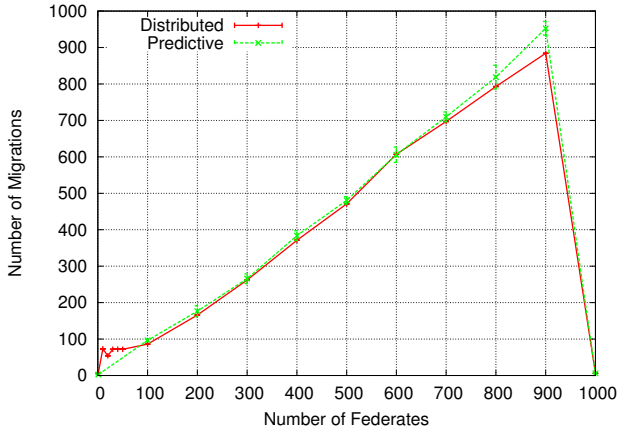


(b)

Figure 2: Performance Analysis for an Increasing Number of Federates with Static Load



(a)



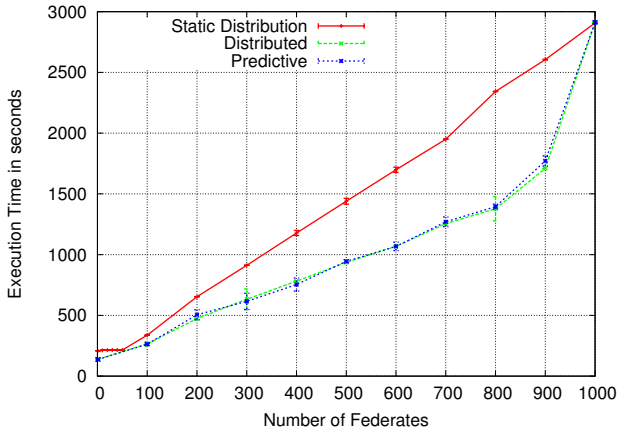
(b)

Figure 3: Performance Analysis for an Increasing Number of Federates and External Background Load

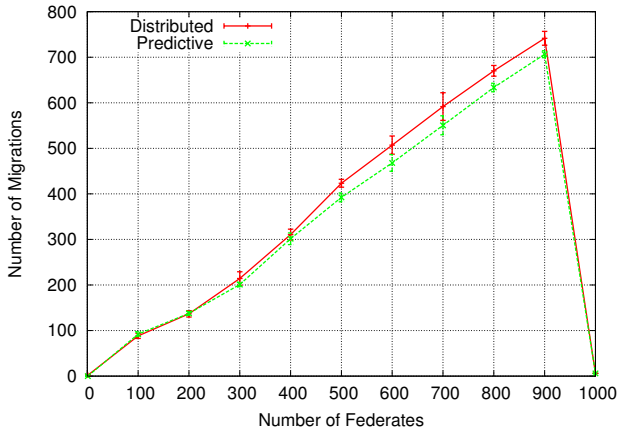
every server, Globus Toolkit 4.2.1 was set up to support the balancing system, and HLA platform with RTI version 1.3 was used to coordinate the experimental simulations

For both balancing systems evaluated in the experiments, their balancing elements were deployed on all computing servers, and the CLBs were placed on each cluster management server. The baseline, as well as the initial configuration of each simulation, consisted in evenly deploying federates on the 55 shared resources. One of the servers was totally dedicated to running the HLA RTI executive. The simulations used in the experiments comprised the coordination of training operations in a two-dimension routing space. In the simulation scenario, 1 to 1000 federates controlled the movement of objects (tanks) during 100 time steps. Minimizing the communication influence on the simulation, the federates calculated the movements of tanks through highly intensive computational tasks.

In this first experimental scenario, federates with static load were deployed on all resources to evaluate the performance gain of the proposed balancing scheme. As shown in Figure 2a, both schemes produce similar performance improvement when comparing their simulation execution times with the baseline for an increasing number of federates. However, when the number of migrations of both schemes are compared, a slight difference appears, as depicted in Figure 2b. In this graph, the predictive scheme produces more migrations than the distributed approach. This increase in the number of migrations was expected because of the changes for the predictive redistribution algorithm. By its definition, the predictive balancing is expected to increase the number of migrations, since 2 other migration opportunities (medium and long term predictions) are available



(a)



(b)

Figure 4: Performance Analysis for an Increasing Number of Federates with Dynamic Load and External Background Process

for every resource. Even though these additional migrations showed a slight decrease of efficiency regarding the ratio performance gain and number of migrations, they enabled the system to prevent some load imbalances.

The experimental test case in Figure 3 described the detection and reaction of the balancing system to external background load. This external load comprises an application that constantly runs and intensively consumes the computational capacity of a resource. Both the predictive and the distributed balancing schemes presented very close experimental results related to performance gain, as shown in Figure 3a. Since the external application was just adding constant load to the system, the application was not able to cause abrupt load variations. Consequently, both balancing systems could detect the presence of the external load and equally respond to imbalances. As presented in Figure 3b, the analysis of the number of migrations also showed the same result as in the previous scenario: the predictive balancing system with larger number of migration moves.

In the last experimental scenario, simulations with load that changed during run-time were employed to evaluate the balancing responsiveness. This behaviour of dynamically changing the load consisted in conditioning simulation federates to produce computational load that varied between high to low intensity on a periodic basis, causing the simulations to exhibit sporadic abrupt load variations. Also, a dynamic external background load was added to the system, contributing to the production of intense load variations in the distributed environment. As depicted in Figure 4a and in the presence of abrupt load oscillations, the predictive balancing scheme demonstrated a slight decrease in simulation time although the performance gain provided by the distributed balancing system reached a level that is hard to be improved for this scenario, as shown in [1]. This increase of performance is a result of the capacity of the proposed system to detect tendencies in the load variations and smooth the abrupt load changes. The distributed balancing scheme was highly susceptible to these variations, producing many unnecessarily precipitated federate migrations, as shown in Figure 4b. Even thought predictive scheme, by design concept, would produce more migrations to enable responsiveness to future trends, the scheme showed a slight decrease in the number of migrations. This difference shows that many unnecessary migrations were avoided during the simulations, presenting a number of migrations larger than the preventive migrations of the proposed scheme.

V. Conclusion

In this paper, a predictive distributed balancing scheme is proposed for redistributing the load of large-scale HLA-based simulations. In the proposed scheme, the monitoring and redistribution mechanisms are modified to enable the introduction of prediction metrics. The predictions contribute to preventing precipitated, unnecessary load changes and allow the detection of future load oscillations trends. The redistribution algorithm evaluates future load trends in three levels of forecasting (short, medium, and long term), allowing the system to react to reasonable transient load changes with short-term predictions and to execute preventive actions with medium and long term predictions. The experiments showed that the proposed predictive scheme produced more federate migrations in static scenarios when compared with the distributed balancing approach. However, in dynamic simulation scenarios, the predictive solution provided a decrease in the number of migrations and a slight increase in performance. The experimental results demonstrated that the proposed balancing system acted as expected, but further work and analysis are required. As future work, other prediction models will be used in the scheme to compare performance gain, methods for directly

detecting cyclic load oscillations will be studied, and a deeper analysis of the balancing behaviour will be performed to improve predictions, calibrate balancing thresholds, and allow the design of dynamic adjustments according to load variations.

REFERENCES

[1] R. E. D. Grande and A. Boukerche, "A dynamic, distributed, hierarchical load repartitioning for hla-based simulations on large-scale environments," in *Proceedings of the International European Conference on Parallel Processing (Euro-Par)*, 2010, to Appear.

[2] B. P. Gan, Y. H. Low, S. Jain, S. J. Turner, W. C. W. J. Hsu, and S. Y. Huang, "Load balancing for conservative simulation on shared memory multiprocessor systems," in *Proceedings of the 14th workshop on Parallel and distributed simulation*. IEEE Computer Society, 2000, pp. 139–146.

[3] M. Y. H. Low, "Dynamic load-balancing for bsp time warp," in *Proceedings of the 35th Annual Simulation Symposium (SS02)*. IEEE Computer Society, 2002, pp. 267–274.

[4] R. Schlagenhaft, M. Ruhwandl, and C. S. H. Bauer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," in *Proceedings of the 9th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1995, pp. 175–180.

[5] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," in *Proc, of the 13th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 101–108.

[6] J. Jiang, R. Anane, and G. Theodoropoulos, "Load balancing in distributed simulations on the grid," in *Proceedings of the International Conference on Systems, Man and Cybernetics*. IEEE Computer Society, 2004, pp. 3232–3238.

[7] P. Peschlow, H. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *Proceedings of the 21st Workshop on Parallel and Distributed Simulation (PADS07)*. IEEE Computer Society, 2007, pp. 219–228.

[8] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary, "Scheduling critical channels in conservative parallel discrete event simulation," in *Proceedings of the 13th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 20–28.

[9] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," in *Proceedings of the 8th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1994, pp. 164–172.

[10] A. Boukerche, "An adaptive partitioning algorithm for conservative parallel simulation," in *Proceedings of the 15th International Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2001, pp. 133–138.

[11] E. E. Ajaltouni, M. Zhang, A. Boukerche, and R. E. D. Grande, "An adaptive dynamic load balancing technique for grid-based large scale distributed simulations," *Journal of Interconnection Networks*, vol. 10, no. 4, pp. 391–419, 2009.

[12] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "An adaptive load balancing middleware for distributed simulation," in *Workshop on Middleware and Performance (WOMP)*, 2006, pp. 864–872.

[13] R. E. D. Grande and A. Boukerche, "Dynamic partitioning of distributed virtual simulations for reducing communication load," in *Proceedings of the IEEE International Workshop on Haptic Audio visual Environments and Games (HAVE)*. IEEE Computer Society, 2009, pp. 176–181.

[14] ——, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," in *IEEE Symposium on Computers and Communications*, I. C. Society, Ed., 2010, pp. 1109–1114.

[15] D. W. Glazer and C. Tropper, "On process migration and load balancing in time warp," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 318–327, 1993.

[16] C. Burdorf and J. Marti, "Load balancing strategies for time warp on multi-user workstations," *The Computer Journal*, vol. 36, no. 2, pp. 168–176, 1993.

[17] C. D. Carothers and R. M. Fujimoto, "Background execution of time warp programs," in *Proceedings of the 10th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 1996, pp. 12–19.

[18] G. Tan and K. C. Lim, "Load distribution services in hla," in *Proceedings of 8th IEEE Distributed Simulation and Real-time Applications*. IEEE Computer Society, 2004, pp. 133–141.

[19] L. F. Wilson and W. Shen, "Experiments in load migration and dynamic load balancing in speedes," in *Proceedings of the 1998 Winter Simulation Conference*. IEEE Computer Society, 1998, pp. 483–490.

[20] E. Deelman and B. K. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems," in *Proceedings of the 12th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1998, pp. 46–53.

[21] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS97)*. IEEE Computer Society, 1997, pp. 32–37.

[22] J. Luthi and S. Grossmann, "The resource sharing system: dynamic federate mapping for hla-baseddistributed simulation," in *Proceedings of the 15th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 2001, pp. 91–98.

[23] W. Cai, S. J. Turner, and H. Zhao, "The resource sharing system: dynamic federate mapping for hla-based distributed simulation," in *Proceedings of the 6th International Workshop on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2002, pp. 7–14.

[24] K. Zajac, M. Bubak, M. Malawski, and P. Sloot, "Towards a grid management system for hla-based interactive simulations," in *Proceedings of the 7th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Comp. Society, 2003, pp. 4–11.

[25] H. Avril and C. Tropper, "The dynamic load balancing of clustered time warp for logic simulation," in *Proceedings of the 10th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 1996, pp. 20–27.

[26] E. E. Ajaltouni, A. Boukerche, and M. Zhang, "An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure," in *Proceedings of the 12th 2008 International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 61–68.

[27] A. Boukerche and R. E. D. Grande, "Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems," in *Proceedings of the 13th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2009, pp. 175–183.

[28] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[29] A. Boukerche and R. E. D. Grande, "Optimized federate migration for large-scale hla-based simulations," in *Proceedings of the 12th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 227–235.

[30] Z. Li, W. Cai, S. J. Turner, and K. Pan, "Federate migration in a service oriented hla rti," in *Proceedings of the 11th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2007, pp. 113–121.

[31] E. S. G. Jr., "Exponential smoothing: The state of the artpart ii," *International Journal of Forecasting*, vol. 22, no. 4, pp. 637–666, 2006.