# Federate Migration Decision-Making Methods for HLA-Based Distributed Simulations *

Raed Alkharboush, Robson Eduardo De Grande and Azzedine Boukerche
PARADISE Research Laboratory
University of Ottawa
Email: {ralkh092,rdgrande,boukerch}@uottawa.ca

***Abstract*—HLA-based distributed simulations tend to suffer from load imbalances and degradation in performance as a result of running on distributed environment. High-Level Architecture (HLA) is a general purpose framework that eases the implementation of distributed simulations on top of dedicated resources without worrying about the computing infrastructure. Due to the high cost of hardware and other factors, some companies have ditched the concept of dedicated resources and shifted towards shared ones which revealed some HLA weaknesses, out of which, dynamic reaction to load imbalances and managing federates on the shared resources. Therefore, different efforts have proposed numerous dynamic load balancing systems to offer a balancing feature to running distributed simulations. In order to perform the load balancing task, these proposed systems gather and make use of a number of simulation and load metrics. Load prediction is a metric that is computed to provide load projections and prevent any prospective load imbalances by migrating federates from an overloaded shared resource to an underloaded shared resource. This work touches the federate migration decision-making process, which is the last step of the balancing task. The proposed federate migration decision-making methods are to overcome the dependency on predefined thresholds in previous work and offer dynamic decisions to migrate federates.**

***Index Terms*—High Level Architecture; Load Balancing; Fuzzy Logic; Impact**

## I. Introduction

Complex distributed simulations, such as largely HLA-based simulations, is one of the topics that have been the focus of many researchers as they provide a fast, yet easy, implementations for solving complicated problems. Because of the dependency of these kinds of simulations on distributed and parallel/concurrent systems, their performance can be easily affected by different factors, such as dynamic changes of simulation load, heterogeneity of resources, presence of external load, and improper distribution of simulation entities. One way to resolve the improper distribution of simulation entities is to initially place these entities, based on the heterogeneity of the resources, in a way to ensure that the distributed simulation would start with a balanced state. Some load balancing management systems that implement the previous solution usually lack the dynamic load redistribution capabilities after the distributed simulation starts executing. Therefore, a number of load balancing management systems have been proposed to dynamically reallocate simulation entities between resources once load imbalances are detected by continually monitoring the load of the resources.

High-Level Architecture (HLA) is a battery-included framework that aims to ease the implementation of distributed simulations. Reusability and interoperability of simulation entities are the concepts HLA introduced to enhance, unite, and simplify designing distributed simulations. HLA allows the reuse of simulation entities (federates) in other distributed simulations (federations), while following policies for exchanging data to allow interoperability. In order to apply these two concepts, HLA forces a set of *rules* to be complied by federates and federations. Moreover, *Interface Specifications* are defined to unite the communication among federates and their respective federations and *Object Model Templates* to publish their objects and interactions to others in the distributed simulation.

The design of HLA was based on usage with dedicated resources, as a result HLA has shown some weaknesses with the implementation of distributed simulations on shared resources, such as inability to control the federates and a proper federate migration protocol to reallocate federates without stopping the distributed simulation. Furthermore, HLA lacks load balancing capabilities. Different proposed load balancing management systems have been proposed to provide solutions to load imbalances while running distributed simulations on shared resources. These systems use different metrics to reallocate federates between resources to balance the simulation environment, such as migration load [1], communication load [2], and projections [3].

The work presented here discusses different dynamic federate migration decision-making approaches . These proposed approaches aim to remove the dependency of a previous work [3] on predefined thresholds which resulted in a load balancing system that is not adaptive to different kind of data or prediction methods. Instead of using static thresholds, the proposed approaches make use of a set of features that are extracted from the current load of the resources to dynamically adjust themselves at every balancing cycle.

This paper is structured as follows. Section 2 presents related work. Section 3 deliberates the incorporated system. Section 4 discusses the proposed federate migration decision-making approaches. Section 5 compares the performance of the proposed approaches against the original approach. Section 6 summarizes the outcome and discusses future work.

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

## II. Related Work

Load balancing becomes a necessity when the execution time matters. As a result, many different dynamic load balancing systems and schemes have been proposed to offer better performance. Some of these designed schemes migrate federates from overloaded resources to underloaded resources based on the communication characteristics and the dependencies between federates. Other systems make use of computational aspects of the distributed simulation to reallocate the load. Other systems make use of prediction models to project the load of the resources and redistribute the load to prevent load imbalances.

*Communication-based* dynamic load balancing systems identify simulation entities that degrade the performance and execution time of the distributed simulation by analyzing the simulation look-ahead and communication dependencies. Look-ahead allows the detection of simulation entities that delays the simulation by affecting other simulation parts [4] [5]. On the other hand, monitoring communication rate enables identifying delays that are triggered by communication latencies. Evaluating the communication dependencies helps in reducing the network distance between parts, and this process has been performed statically [6] [7] and dynamically [5] [8] [9]. Using Communication-based load balancing system could be beneficial, however, it lacks the capability to deal with load imbalances in shared resources.

*Computational-based* systems analyze the computational load of the federates or the resources in order to migrate federates from overloaded resources to underloaded resources. Different load balancing systems have been proposed that aim to mainly enhance the performance of simulation execution time by improving the execution time of each federate [10] [11] [12], while other systems redistribute loads between the shared resources to ensure even allocation [13] [14] [15]. The previous systems lack proper load reallocation with the presence of external background load and dealing with resources with different configurations. As a result, some enhanced solutions have been proposed [16] [3] to overcome these limitations.

*Prediction-based systems* use different forecasting methods to project the load of the shared resources in a number of future balancing cycles. These projections are categorized into two categories, where each category serves the system in a specific way. The first kind of projection recognizes load imbalances and reallocates loads to prevent these load imbalances. On the other hand, the second kind of projections help the load balancing system to identify any future load imbalances so the system would react early towards these imbalances and perform precautional procedures. These systems depend on the accuracy of the forecasting methods [3] [17].

The approach in [3] [2] uses thresholds to decide if a load migration is needed. These thresholds are predefined based on a certain case, which makes this approach not adaptive to new forecasting methods. Therefore, different adaptive federate migration decision-making approaches are proposed.

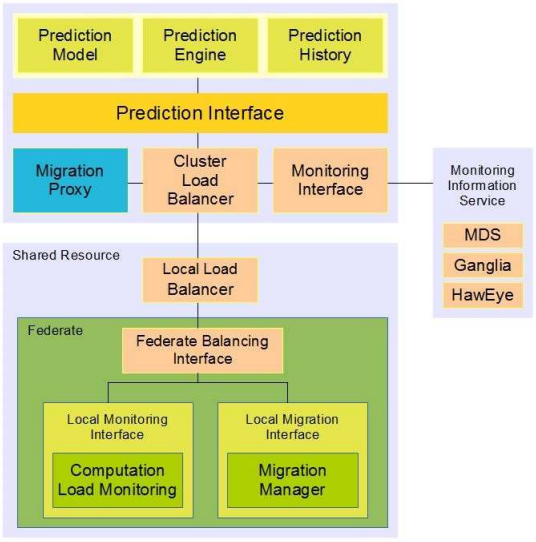



Fig. 1: Prediction-Based Dynamic Load Balancing Architecture

## III. Predictive Load Balancing Scheme

The proposed federate migration decision-making approaches are incorporated into a dynamic prediction-based load balancing management system described by De Grande and Boukerche [3] and presented in Figure 1. The process of load balancing in the system goes over two steps: monitoring the shared resources to identify the overloaded and underloaded resources, and applying certain approaches to decide whether a migration is needed from an overloaded to an underloaded resource.

In the incorporated system, resources with similar configurations are grouped into one cluster and managed by a Cluster Load Balancer (CLB). Each CLB is connected to Local Load Balancers (LLBs) that reside in the shared resources. These connections allow CLBs to manage the resources that belong to it and the federates that run on them. CLBs are connected to its neighboring CLBs so that they can exchange information about their own environment. CLBs gather load information about its resources by communicating with the Monitoring Interface and then, with the help of the Prediction Interface, analyze the candidate list of resources to reallocate and evenly balance their loads.

CLBs access monitoring related data by communicating with the Monitoring Interface, which extracts load data from sending inquiries to Monitoring Information Services (MIS). Once an inquiry is received by MIS, MIS would send only the load related information of resources under the management of the inquiring CLB. MIS uses Grid services, through monitoring and discovery services, to access such information. Grid is a management system for shared resources that is used to coordinate the execution of distributed applications [18].

Each resource has an LLB that gathers information about the load of the simulation entities that run on it and response to queries and commands sent by its managing CLB. Collecting

Load information is performed upon request from the CLB, where the receiving LLB forwards the collecting request to the Federate Balancing Interface. Federate Balancing Interface then forwards the request to the Local Monitoring Interface of each federate it hosts to retrieve the CPU consumption of the hosted federates. LLB forwards migration calls from its CLB to its Migration Manager.

Similarly to [19] [20], the load balancing system performs federate migrations in two steps: transmitting static data and initialization files through Grid Services from the overloaded resource to the underloaded resource, followed by sending the execution status and queued incoming messaged through the Migration Manager. Migrating federates between two resources that reside in different clusters are conducted through a Migration Manager.

As each Prediction Model requires different kind of historical data, Prediction History keeps a copy of previous data that are needed by the implemented Prediction Model. The Prediction Engine processes the Prediction Model on the data it receives from both CLB and Prediction History. In the incorporated system, a *smoothed value* and a *trend* are kept in the Prediction History as the system uses an extension of Exponential Weighted Moving Average, named Holt's Model.

### A. Re-distribution Algorithm

In order to enable the balancing capabilities in an HLA-based distributed simulation, the incorporated system uses a redistribution algorithm to balance the load among the shared resources. The algorithm consists of three different stages: monitoring, reallocation, migration.

The dynamic load balancing management system continually monitors the shared resources which enables the responsiveness to any sudden load oscillations. The monitoring stage is periodically executed every $\Delta$t, which is constrained by the monitoring tool's refresh rate. To ensure accurate projections, the system first filters the gathered load information to eliminate abnormalities. Once filtered, the system normalizes the data to ensure a fair comparison between the resources. The system used the set of normalized loads along with the previous history and the migration status to calculate the projection of each shared resource in three different future balancing cycles: short-term, medium-term, and long-term projections. The values of the projections represent the predicted load of a resource in a specific future balancing cycle.

The load balancing system applies a pair matching algorithm, which is shown in Algorithm 1, between the resources in each balancing cycle. The system gives more importance to the balancing cycles that are closer to the current cycle. Before applying the pair matching algorithm, the system sorts the list of resources according to their loads. The system then starts the pair matching process against the sorted list by adjusting a list of parameters, $min$, $\delta$ and $\theta$, to be used for calculating thresholds upon matching each pair of resources. The first threshold, $min \times \delta$, is to ensure that the difference of the load between the overloaded and the underloaded resources is large and justifies a federate migration. The second threshold, $min \times \theta$, is larger that the first threshold and it is only applied when the overloaded resource has one federate which ensures that the load of the overloaded is abnormal and transferring a federate can lower its load. Once the thresholds' values are exceeded, the pair matching algorithm triggers a migration to reallocate the federate with the lowest load from the overloaded resource to the underloaded resource.

Based on the success rate of the local migrations, each CLB starts the process of inter-domain migrations, migrations to other clusters. The CLB starts by requesting Cluster Load, the average load of its shared resources, from its neighboring CLBs. When the Cluster Load is received by the requesting CLB, a process starts to recognize local overloaded resources, when compared against the load of the resources of the other cluster, by identifying load imbalances between the shared resources of the involved clusters. This list of local overloaded resources is sent to the neighboring CLBs for projection calculations and pair matching. When the neighboring CLBs conclude the process and finalize the list of possible migrations between themselves and the other cluster, the list of migrations are sent to the other cluster to start the inter-domain migrations.

**Algorithm 1** Local Prediction Pair-Match Evaluation Algorithm

**Require:** $srcRsc, dstRsc, min, \theta, \delta$
  $min, \delta, \theta \leftarrow adjustParameters(srcDirection, dstDirection, type)$
  **if** $dstLoad < min$ **then**
    **if** $numberFederates(srcRsc) > 1$ **then**
      $create\_migration\_move(srcRsc, dstRsc)$
    **else**
      **if** $numberFederates(srcRsc) \geq 1 \& srcLoad > (min \times \theta)$ **then**
        $create\_migration\_move(srcRsc, dstRsc)$
      **end if**
    **end if**
  **else**
    **if** $(srcLoad - dstLoad) > (min \times \delta)$ **then**
      **if** $numberFederate(srcRsc) > 1$ **then**
        $create\_migration\_move(srcRsc, dstRsc)$
      **else**
        **if** $numberFederate(srcRsc) > 1 \& srcLoad > (min \times \theta)$ **then**
          $create\_migration\_move(srcRsc, dstRsc)$
        **end if**
      **end if**
    **end if**
  **end if**

### B. Forecasting Load Status

The load balancing computes three load forecasting prediction model independent projections: short, medium and long term. Each of these projection serves a certain goal towards balancing the load of the shared resources. The short-term projection (SP) helps the system react to current load imbalances. On the other hand, medium-term and long-term projection, MP and LP, identify future load imbalances which are used to help the balancing system to be proactive and prevent any future load imbalances. The projections represent the predicted load of a resource at a certain balancing cycle in the future. Short, medium, and long term projections are defined as 1, 3, and 5 balancing cycles, respectively.

### C. Prediction Models

The computational load of each shared resource is collected in a list in time fixed intervals. This list represents the load behavior of each resource in time. Thus, time series prediction methods are used in order to forecast loads of the shared resources.

The system uses a time series forecasting model, namely Holt's model, to calculate the projections as the collected load information represents the behavior of the resources in time. Holt's model is an extension to the well-known double Exponential Weighted Moving Average (EWMA). EWMA builds a relationship between the internal elements of the provided data in three different aspects [21]. Single smoothed EWMA computes a smoothed value of the predicted term based on an exponentially averaging technique. Double EWMA uses the trend of the data to compute the projection. Triple EWMA divides the data into seasons, and uses this knowledge along with the trend to compute the projection. Through experiments, the data has shown a trend but has not shown any sign of seasonality.

As Holt's model is the model implemented in the Prediction Model, the Prediction Engine computes the forecasting value, $F_{m+i}$ at a future cycle $i$ from point $m$, as represented in Equations 1, 2, and 3. Using the monitoring related data, provided by the Monitoring Interface through CLB, along with the historical data, which is stored in the Prediction History, the Prediction Engine starts by computing the smoothed value, $sum_i$ based on the current actual load of a resource, $elem_i$, the previous smoothed value, $sum_{i-1}$, and the previous trend, $t_{i-1}$. Computing the trend, $t_i$ enables the extrapolating of the average of the smoothed value and is based on the tendency, $sum_i - sum_{i-1}$, and the previous trend, $t_{i-1}$,. The sign of the tendency represents the direction of the load, increasing or decreasing, while the value of the tendency shows how steep the trend is. Once the smoothed value and the trend are computed, the Prediction Engine calculated the three different projections: SP, MP, and LP, as shown in Equation 4.

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

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

$$F_{i+m} = sum_i + m \times t_i, m \in \{1, 3, 5\} \quad (3)$$

$$SP = F_{i+1}, MP = F_{i+3}, LP = F_{i+5} \quad (4)$$

At the initial stage, the Prediction History sets all previous smoothed value to $0$ and the first smoothed value to $elem_0$. This is to ensure that the projection calculations will not crash when previous history are not presented. Similarly, the Prediction History assigns the value of $0$ to the first trend, $t_0$, as it has no idea of the tendency of the load.

## IV. Federate Migration Decision-Making

Two different thresholds were discussed that the system uses in order to decide if there is a need to perform a load migration. These thresholds are considered dynamic but not adaptive, as their values are changed to predefined values based on the current type of projection and the tendency of both the overloaded and underloaded resources. These predefined values are set after running a set of experiments and the best set of thresholds had been chosen. As a result, any new change or new improvement to the system would require running a set of experiments to produce a new set of thresholds. This process would be time consuming and inefficient. Therefore, the next section proposes different federate migration decision-making approaches that are both dynamic and adaptive.

### A. Restricted Approach

The Restricted Approach (RA) replaces the dependency on the predefined thresholds with a rule-based Expert system. RA uses the load and direction of the source and destination resources, the mean of all resources in the current balancing cycle, and a computed tolerance. The tolerance is a value proposed in the original approach that defines a range that once used along the mean, the dynamic load balancing system can define a *balanced* area where resources with loads that reside in that area are considered balanced. RA uses the concept of tolerance to compute a region around the quantitative value of the resource's load to define a *resource area*, as shown in Figure 2.

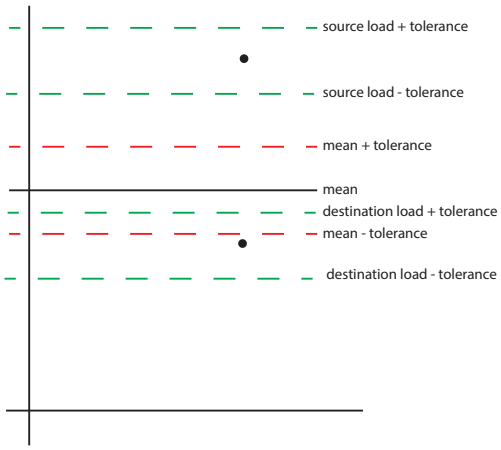



Fig. 2: RA Areas

As RA is an expert system, a set of rules are defined based on the observation of when the migrations should be triggered. The rules are based on whether there exists an intersection between the *balanced area* and the *resource area*. The existing of such intersection raises a *balanced flag* as the resource is near the balanced region and there is no justification to have it involved in a federate migration. A resource is considered *unsafe* and marked for possible migration when its resource area does not intersect with the balanced area.

RA can be considered as a loose approach as it triggers a load migration almost all the time except when both resources are not safe, the destination has an increasing load tendency, and the source has either a decreasing or stable load tendency. In simple words, both resources are heading towards the

balanced area. Thus, issuing a federate migration from the overloaded source to the underloaded destination would end up in the possibility of changing the destination's status from underloaded to overloaded in the few next balancing cycles. Additionally, the previous rule prevents issuing migration calls in the future when overloaded resources become underloaded resources.

### B. Direction Ratio Approach

RA uses a loose set of rules that can cause inconsistency in the federate migration process. Thus, a more restricted approach was needed. The restrictions to be applied in the new approach must limit the number of migrations to only those that would eventually lead to a balanced environment. In order to do that, Direction Ratio (DR) adds *Ratio* to the list of metrics used by RA. The ratio represents the percentage of source's load tendency to the destination's load tendency.

In addition to RA rules, DR applies more limitations to the number of triggered migrations by analyzing the direction of the candidates' loads. Examining the different possibilities of the directions gives an insight on whether a migration has the possibility of affecting the load and status of both candidates. The ratio is used as a metric to calculate the effect factor. The ratio is then compared to a fixed value, and based on the comparison, DR either triggers or ignores the migration. Once DR hits one of the following rules, it triggers a migration. Each rule is represented by set of conditions that are colon separated. The first and third fields represent whether the Source/Destination are safe (S) or not (N). The second and fourth fields signify the tendency, increasing ($\nearrow$), decreasing ($\searrow$), or not used ($-$), for the source and destination, respectively.

- S:$-$:S:$-$
- S:$\nearrow$:N:$\searrow$
- S:$\nearrow$:N:$\nearrow$, Ratio $> 0.5$
- S:$\nearrow$:N:$\searrow$, Ratio $< 0.5$
- N:$\searrow$:S:$\searrow$, Ratio $< 0.5$
- N:$\nearrow$:S:$\searrow$
- N:$\nearrow$:S:$\nearrow$, Ratio $> 0.5$

### C. Fuzzy Approach

A different idea was desired to increase the balancing convergence speed of the shared resources. This led to start investing a new approach that identifies different kinds of loads and reacts differently for each type of load. The idea is based on categorizing the loads into sets that represent the level of load the resource is under. Instead of a federate migration decision-making approach that depends on comparing different metrics, the desired approach would make use of the resources' categories to trigger a number of federates to insure a quick convergence speed. Fuzzy Logic is an approach that works by categorizing inputs and provides an output based on predefined rules. Fuzzy Approach (FA) assigns each load to a set, then goes over a number of predefined rules to see which one is applied to the current situation and then perform the rule's actions.

The approach has 6 predefined sets, where a pair of 3 sets are dedicated to cover each status, overloaded and underloaded. The 3 sets per status covers different ranges of the loads: *extreme*, *normal*, and *low*. The combination of the status with their ranges of loads results in the following fuzzy sets: *extremely overloaded*, *overloaded*, *slightly overloaded*, *slightly underloaded*, *underloaded*, *extremely underloaded*.

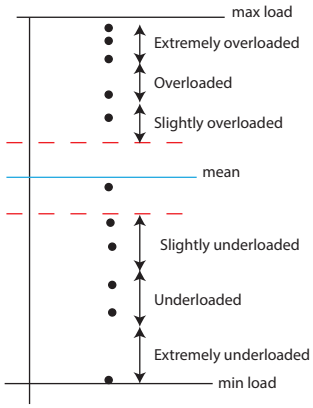


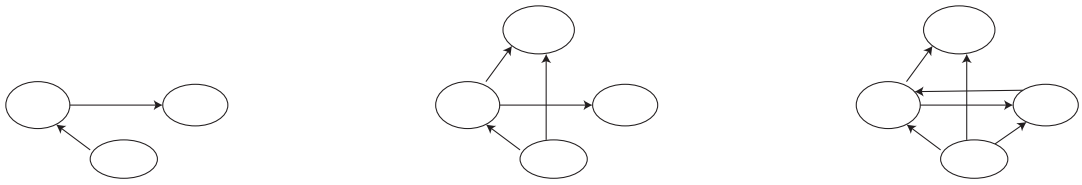

Fig. 3: Fuzzy Approach: Dividing Sets

Based on these sets, FA goes through the rules that define the number of migrations to perform from the overloaded to the underloaded resource. In this case, FA migrates 2 federates when the source is extremely overloaded and the destination is extremely underloaded. However, it does not trigger any migrations when it feels that issuing a migraion call would cause load imbalances between resources, as when the source and the destination match any of these sets, respectively, (extremely overloaded, underloaded), (overloaded, extremely underloaded), and (slightly overloaded, slightly underloaded). In the other cases, FA triggers only one migration.

At each balancing cycle, FA creates 3 equal sets for each status that ranges from the min/max to the balanced region, as shown in Figure 3. A loop starts by selecting candidates and assigns each of the candidates to their respective category. FA then evaluates the appropriate rule. Once a rule is evaluated, the system will transfer the proposed number of federates by FA from the source to the destination.

### D. Impact Approach

All of the proposed approaches, including the approach that is used in the original approach, remove the the candidates from the candidate list once a migration is triggered. When the resources are removed from the candidate list, the balancing system would not consider analyzing any possible load migrations in the current balancing cycle or the next one for the removed candidates. Because of this behavior, the balancing system does not take into account any other possible scenarios, that involve the removed resources, which could be more beneficial to the shared resources. Thus, Impact Approach (IA) is designed to propose analyzing possible migrations and issue the most beneficial ones.



(a) SP migrations (b) MP migrations (c) LP migrations

Fig. 4: IA migrations

Impact Approach (IA) takes into account all possible scenarios any resource is involved in and triggers the most valuable migrations. In order to do so, IA computes an *impact factor* for each possible load migration, represented by

$$Impact = \frac{srcLoad - mean}{dstLoad} \tag{5}$$

The mean is subtracted from the the load of the source to bring the source's load closer to the destination's load. The impact factor's value represents the impact degree of the migration to the shared environment. The higher the value is, the more important it is to trigger this migration.

For this approach to work, it goes first over all the possible migrations for the short term projections. The possible migrations are presented to the system as a unidirectional graph, from an overloaded source node to an underloaded destination node, and the impact factor as the weight of the edge, as shown in Figure 4. Then, the same candidate list is sent to be processed for medium term projections. The possible list of migrations is added to the unidirectional graph. When IA finishes analyzing the possible load migrations for the medium term projections, the candidate list is sent untouched to be evaluated for long term projections. Similarly, the list of migrations are added to the unidirectional graph.

Once all possible migrations are computed, IA goes over through the graph and first perform the migration with the highest impact factor. Once it issues a high impact migration, IA removes the candidates of the involved migration and their respective edges from the graph, eliminating any migration from or to the candidates in the current balancing cycle. The process continues until no more edges in the graph. The remaining resources are nominated as candidates for the next balancing cycle.

## V. Experimental Results

The proposed federate migration decision-making approaches are compared to the original decision-making approach [3]. The evaluation process was conducted in two different stages. The first stage analyzes the *behavior* of the proposed approaches on data samples, while the second stage compares the performance of the proposed approaches against the original approach.

*1) First Stage:* : The behavior to be examined is the distribution of the migrations triggered per each projection in a single balancing cycle. Figure 5 shows the behavior for such output per each of the proposed federate migration decision-making approach.

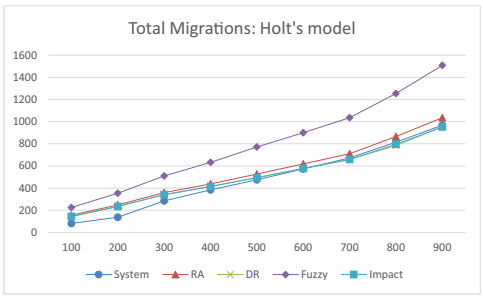
By looking into the total amount of migrations, the approaches have the same behavior. However, FL exceeds the other approaches. The reason of the high number of migrations lays in how FL categorizes the candidates. When FL categorizes the overloaded resource as an *extremely overloaded* and the underloaded resource as an *extremely underloaded*, FL transfers two federates from the overloaded to the underloaded instead on one. The previous rule does play an important rule towards increasing the convergence speed to have a balanced environment. However, its drawback is that it triggers more migrations.

This rule is particularly applied the most in these results due to the way FL is implemented. FL *always* creates the *extremely overloaded* and *extremely underloaded* that are based on the *current* loads instead of the *overall* load of the system. In a nutshell, FL creates the extremely overloaded set even when the load is not considered extremely overloaded for the shared resources as a whole, however, it represents an extremely overloaded load for the list of loads in the current balancing cycle.

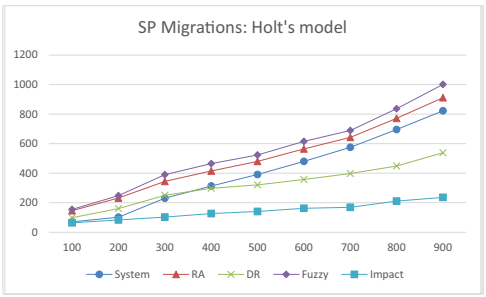
To better understand the behavior of the proposed approaches, the number of migrations triggered for SP, MP and LP was analyzed. By comparing the results of the approaches with the results of the original approach (System), RA produces more SP migrations. This is justified by the loose rules we have defined in RA. The more loose the rules are, the more migrations are to be triggered. Because of the high number of migrations initiated in SP phase, the less candidates are sent to be processed in MP phase. The effect of having low number of candidates is seen in the MP migrations of RA compared to the original where the number of migrations in RA is less than the original system. However, running simulations with more federates results in more inconsistencies, where the looseness is taken advantage from and RA starts producing more migrations than the original system. After this stage, RA has consumed most of its candidates results in low migrations initiated in LP.

However, DR applies more restrictions and this resulted in reducing the number of migrations in SP. This allows for more candidates to be sent for MP phase. At this stage, DR has provided MP phase with the highest number of candidates, therefore, more MP migrations than the original system and RA. Eventually, sending more candidates to LP.

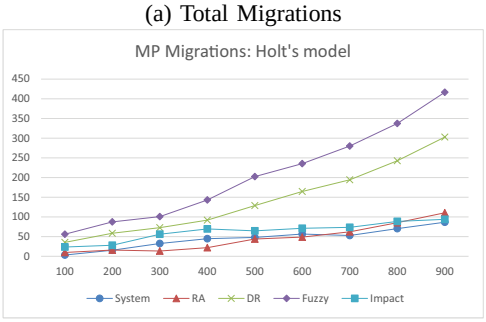
IA shows the lowest number of migration triggered in SP.



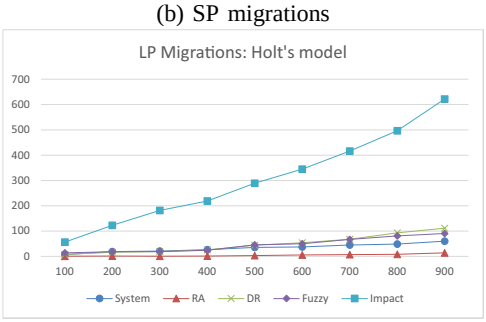

(a) Total Migrations




(b) SP migrations




(c) MP migrations




(d) LP migrations

Fig. 5: Migrations per approach

In a previous study [17], we can see that the SP has the lowest projections among the rest (MP and LP). As a result, The impact factor computed for SP is smallest and LP is the highest. From how the IA works, IA gives more importance to migrations with the highest impact factor first. Therefore, the number of migrations triggered in LP is the highest, followed by MP, and then SP. This shows that the migrations are issued in a manner where LP migrations are given more importance than MP, which is higher than SP.

Because of the implementation of the rule where two federates are transferred, FL produces the highest number of migrations in SP. It is true that the number of candidates sent for MP processing is low at this stage, however, the rule once again is applied, at least once, which results in a high number of migrations in MP. At LP stage and because most of the rules initiate a migration, the number of candidates is lower than the rest. The two-migration rule again plays an important role here as it increases the number of initiated migrations.

*2) Second Stage:* : The goal of the second stage is to analyze the gained performance of the proposed approaches. Moreover, this analysis would help identifying the suitable use cases for the different combinations.

To properly evaluate the approaches, a heterogeneous testbed was used that consisted of two clusters of computing servers: DELL and IBM. The DELL cluster contains 15 computing servers interconnected through a Myrinet optical network that allows for data transmission up to 2 gigabits per second; also each computing server contains a Quadcore 2.40GHz Intel Xeon CPU and 8GB of DIMM DDR RAM. The IBM cluster was composed of 23 computing servers interconnected by a gigabit Ethernet network; each server contains a Core 2 Duo 3.4 GHz Intel Xeon CPU and 2GB of DIMM DDR RAM. The management nodes of the two clusters are interconnected with a Fast-Ethernet link. Both clusters run on Linux and use Globus Toolkit 4.2.1 and HLA platform with RTI version 1.3 to support the load balancing systems and coordinate the experimental results. The results below are the average of 6 runs. Tables I and II shows the rounded average number of migrations and the execution time of each approach, compared with the original approach, with confidence rate at 95%.

From Table I, there are three approaches that give the best results, but in different federates configurations. With low federate configurations, 100 to 300 federates, the original approach triggers the least number of migrations. In such configurations, the load of the shared resources, in general, is low, thus the differences between the overloaded and the underloaded resources are low and do not always reach to the the predefined thresholds in SP. Therefore, losing some migrations would yield to a low number of migrations.

Looking at the low federate configurations, IA scores the second in the lowest number of migrations as it performs the most beneficial migrations and ignores the ones that does not help balancing the environment. Both RA and DR have loose rules, which at this point does not reduce nor limit the triggered migrations as the system does not face serious load imbalances. However, when we apply one of the medium federate configurations, federates from 400 to 600, we notice that the load differences between the overloaded and under-loaded resources become larger. In the original approach, this larger difference would go closer to, or beyond, the predefined thresholds and as a result, the original approach would issue

TABLE I: Number of migrations per approach

| Feds | Org | RA | DR | FA | IA |
|---|---|---|---|---|---|
| 100 | 114(17) | 164(12) | 156(16) | 229(24) | 152(13) |
| 200 | 176(29) | 278(13) | 257(9) | 389(11) | 249(5) |
| 300 | 306(10) | 373(9) | 364(9) | 552(17) | 342(11) |
| 400 | 423(11) | 451(14) | 436(25) | 695(9) | 414(17) |
| 500 | 493(31) | 537(20) | 510(10) | 815(17) | 489(6) |
| 600 | 589(36) | 629(9) | 586(13) | 945(21) | 561(18) |
| 700 | 679(38) | 707(14) | 643(23) | 1078(26) | 658(18) |
| 800 | 745(26) | 781(31) | 685(21) | 1181(26) | 743(22) |
| 900 | 833(50) | 873(25) | 784(34) | 1339(14) | 845(55) |

more migrations. At this point, the rules of DR become more strict as the system starts facing more load imbalances. In high configurations, from 700 to 900 federates, HLA produces high projections for MP and LP, which is explained in [17]. As DR covers, in its design, more load scenarios than the rest of the approaches, it rejects unnecessary migrations that are triggered by other approaches, resulting in less overall migrations. DR's strict rules, along with the dependency on the Ratio, play an important rule in selecting the most suitable migrations.

For the execution time, as shown in Table II, the original approach is the fastest among the approaches. Throughout analyzing the number of migrations performed at each stage, we found that the number of inter-domain migrations was larger in the original approach. Thresholds in the original approach limit the number of local migrations, however, it is more loose when it comes to inter-domain migrations. Inter-domain migrations help the system to reallocate loads from low end clusters to high end clusters, which would help processing the simulation faster. On the other hand, as the proposed approaches adapt themselves to data at each balancing cycle, they give more importance to local migrations, thus losing the benefit of utilizing the powerful clusters.

## VI. Conclusion and Future Work

In this paper, a number of federate migration decision-making approaches are proposed for a load reallocation on a Large-scale HLA-Based distributed simulations. The approaches are the result of finding a solution to a limitation of a current load balancing management system. RA and DR introduce rules that are independent to any fixed value. Both approaches cover different set of rules and present different flexibilities. FA assigns resources to sets and compares the combination against predefined rules to trigger the required number of migrations for the system to converge faster. IA shows a flexible approach to analyze all possible migrations and trigger the most important ones first through the use of impact factor.

For future work, the results concluded in this work would be compared against SSRT [17] to identify the suitable use case of each combination, an in depth analysis on the complexity of each federate migration decision-making approach is performed to find the hotspots in their algorithms, a dynamic fuzzy logic set generator is designed to built sets based on the overall and historical loads instead of the current load.

TABLE II: Execution time, in seconds, per approach

| Feds | Org | RA | DR | FA | IA |
|---|---|---|---|---|---|
| 100 | 523(54) | 609(40) | 599(30) | 604(43) | 579(31) |
| 200 | 816(19) | 1011(34) | 972(32) | 999(46) | 952(44) |
| 300 | 1212(47) | 1363(19) | 1355(14) | 1355(40) | 1312(52) |
| 400 | 1512(55) | 1653(47) | 1658(63) | 1694(13) | 1577(132) |
| 500 | 1808(63) | 1993(42) | 1958(41) | 1990(52) | 1954(22) |
| 600 | 2139(108) | 2315(35) | 2298(34) | 2312(46) | 2149(123) |
| 700 | 2509(63) | 2627(35) | 2611(62) | 2653(29) | 2586(47) |
| 800 | 2792(27) | 2920(55) | 2907(68) | 2958(64) | 2914(69) |
| 900 | 3192(57) | 3270(29) | 3262(98) | 3346(67) | 3314(96) |

## References


[1] R. De Grande and A. Boukerche, "Dynamic load redistribution based on migration latency analysis for distributed virtual simulations," in *Haptic Audio Visual Environments and Games (HAVE) 2011*.

[2] ——, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," in *Computers and Communications (ISCC)*.

[3] ——, "Predictive dynamic load balancing for large-scale hla-based simulations," in *Distributed Simulation and Real Time Applications (DS-RT) 2011*.

[4] B. P. Gan, Y.-H. Low, S. Jain, S. Turner, W. Cai, W.-J. Hsu, and S.-Y. Huang, "Load balancing for conservative simulation on shared memory multiprocessor systems," in *Parallel and Distributed Simulation (PADS) 2000*.

[5] M. Low, "Dynamic load-balancing for bsp time warp," in *Simulation Symposium 2002*.

[6] R. Schlagenhaft, M. Ruhwandl, C. Sporrer, and H. Bauer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," in *Parallel and distributed simulation (PADS) 1995*.

[7] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," in *Proceedings Parallel and Distributed Simulation (PADS) 1999.*

[8] P. Peschlow, T. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *Principles of Advanced and Distributed Simulation (PADS) 2007*.

[9] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary, "Scheduling critical channels in conservative parallel discrete event simulation," in *Parallel and Distributed Simulation (PADS) 1999*.

[10] D. Glazer and C. Tropper, "On process migration and load balancing in time warp," *Parallel and Distributed Systems (PADS) 1993*.

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

[12] C. Carothers and R. M. Fujimoto, "Background execution of time warp programs," in *Parallel and Distributed Simulation (PADS) 1996*.

[13] L. Wilson and W. Shen, "Experiments in load migration and dynamic load balancing in speedes," in *Simulation Conference Proceedings, 1998.*

[14] E. Deelman and B. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems," in *Parallel and Distributed Simulation (PADS) 1998*.

[15] A. Boukerche and S. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Parallel and Distributed Simulation (PADS) 1997*.

[16] A. Boukerche and R. De Grande, "Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems," in *Distributed Simulation and Real Time Applications (DS-RT) 2009.*

[17] R. Alkharboush, R. De Grande, and A. Boukerche, "Load prediction in hla-based distributed simulation using holt's variants," in *Distributed Simulation and Real Time Applications (DS-RT) 2013.*

[18] I. Foster, "The anatomy of the grid: enabling scalable virtual organizations," in *Cluster Computing and the Grid 2001*.

[19] A. Boukerche and R. Grande, "Optimized federate migration for large-scale hla-based simulations," in *Distributed Simulation and Real-Time Applications (DS-RT) 2008.*

[20] Z. Li, W. Cai, S. Turner, and K. Pan, "Federate migration in a service oriented hla rti," in *Distributed Simulation and Real-Time Applications (DS-RT) 2007*.

[21] E. S. G. Jr., "Exponential smoothing: The state of the art - part ii," *International Journal of Forecasting*, vol. 22, no. 4, 2006.