Load Prediction in HLA-Based Distributed Simulation Using Holt's Variants

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

***Abstract*—Due to the dependency of HLA-Based distributed simulations on the resources of distributed environments, simulations can face load imbalances and can suffer from low performance in terms of execution time. High-Level Architecture (HLA) is a framework that simplifies the implementation of distributed simulations; and, it has been built with dedicated resources in mind. As technology is nowadays shifting towards shared resources, the following two weaknesses have become apparent in HLA: managing federates and reacting towards load imbalances on shared resources. Moreover, a number of dynamic load management systems have been designed in order to provide a solution to enable a balanced simulation environment on shared resources. These systems use some specific techniques depending on certain simulation or load aspects, to perform the balancing task. Load prediction is one such technique that improves load redistribution heuristics by preventing load imbalances. In this work, we present a number of enhancements for a prediction technique and compare their efficiency. The proposed enhancements solve observed problems with Holt's implementations on dynamic load balancing systems for HLA-Based distributed simulations and provide better forecasting. As a result, these enhancements provide better forecasting for the load of the shared resources.**

***Index Terms*—High Level Architecture, Load Balancing, Exponential Weighted Moving Average, Holt's model**

I. Introduction

Distributed simulations, such as large HLA-Based simulations, have been providing easy and fast implementations for different complex problems that many researchers strive to find answers to. Distributed simulations depend on distributed and parallel/concurrent systems; this being so, several factors, which are listed below, can affect the performance of the distributed simulation: improper distribution of simulation entities among the resources, the existence of external load in the resources, resource heterogeneity, and dynamic changes in simulation load. A load balancing system can initially distribute simulation entities based on resource heterogeneity before starting the simulation. This method provides a static way to prevent load imbalances in the beginning of the simulation. However, such systems will not provide a solution to simulations with dynamic load changes that cause load imbalances afterwards. As a result, several dynamic load balancing systems have been designed in order to prevent load imbalances by continually monitoring load. Dynamic Load Balancing improves the performance of distributed simulations by dynamically redistributing simulation entities once a load imbalance is predicted.

High-Level Architecture (HLA) is a fully equipped framework that simplifies the implementation of distributed simulations. HLA introduces two concepts to distributed simulations that improve the design of distributed simulations, reusability and interoperability of simulation entities. Reusability allows simulation entities (federates) to be reused in different distributed simulations (federations), where interoperability adds the capability for those simulation entities to communicate. In order to use the concepts of reusability and interoperability, HLA introduces rules that must be followed by federates and federations. Moreover, HLA defines *interface specifications* to allow for the communication between federates and their respective federations and to allow *Object Model Templates* to publish objects and interactions used in the distributed simulation.

As HLA was built with dedicated resources in mind, HLA experiences weaknesses when it comes to an environment composed of shared resources. HLA, for example, lacks the ability to control distributed simulation entities on shared resources and a federate migration protocol that moves federates around without pausing the whole distributed simulation. Most importantly, HLA is not capable of performing any load balancing function.

Extensive work has been conducted in the field of designing systems to prevent load imbalances in distributed simulations. Such resource balancing systems have used different mechanisms in order to balance different elements in a distributed simulation environment, such as communication load [1], migration load [2], and prediction [3], which uses an extension of Exponential Weighted Moving Average (EWMA), with double exponential smoothing for load forecasting, this is named Holt's model.

This work proposes different modifications and enhancements to the available prediction mechanism: it aims to further understand in depth Holt's prediction method to modify it to have better forecasting of dynamic load changes. The proposed solutions attempt to solve two important flaws that have been identified: Holt's linearity relation between projection values and time intervals, and slow responses to load oscillations.

The paper is organized as follows. Section 2 presents related work. In Section 3, different variants of Holt are described and their algorithms are detailed. Section 4 discusses the outcome of the proposed enhancements. A conclusion is presented in Section 5 along with future work.

II. Related Work

The necessity of load balancing systems arises in distributed simulation environments when a better performance in terms of execution time is required. As application developers strive to make faster simulations, different systems and schemes have been proposed. Some of these systems migrate simulation entities from one shared resource to another, based on the dependency of simulation entities and communication behavior. Other systems concentrate on other aspects, such as the re-arrangement of computational needs for simulation entities. Other systems use prediction models to identify load imbalances in shared resources.

Communication-based load balancing systems aim to recognize simulation entities that decrease the performance of the distributed simulation. The look-ahead mechanism allows for the detection of simulation entities that cause delays to the simulation. By monitoring the communication rate, systems can identify delays that are the result of communication latencies. The analyzing of communication dependencies has been performed statically [4] [5] and dynamically [6] [7]. Communication-based systems add some value to load balancing systems; however, they can not deal with load imbalances in shared resources effectively.

Computational-based load balancing systems migrate the simulation entities between shared resources, based on the computation load of the simulation entity itself or on the shared resource. Different systems [8] [9] [10] focus on improving the performance of execution for each simulation entity. On the other hand, other systems [11] [12] [13] aim to improve the performance of execution per shared resource. These systems show limitations when it comes to resource heterogeneity and the presence of external background load. As a result, other systems have been proposed in order to deal with these limitations: [14] [15] [3].

Some *Prediction-based* load balancing systems [3] project the load of shared resources at certain times, into the future, or more specifically, a number of future balancing cycles. Based on the projections, the system conducts a pair matching algorithm between over-loaded and under-loaded shared resources, this is done to analyze the migrations needed for a balanced distributed simulation. Prediction-based models outperform normal distributed load balancing systems [15] because of the applied measures that forecast the load, using *Holt double exponential smoothing*, and deal with load oscillations. These measures have been proven to prevent unnecessary load redistribution moves, which result in reducing execution time with less migrations.

Holt's double exponential smoothing is an extension of the EWMA commonly used in a number of fields for making predictions; the following represents examples of scenarios in which this model is used: forecasting the textile and garment exports of China [16], predicting the number of cases for diseases [17], compression floating point numbers [18], forecasting the sales of products [19].

Sarimah *et al* [17] compared the performance of 6-time-series exponential smoothing models: Single Exponential Smoothing, Double Exponential Smoothing, Holt's Double Exponential Smoothing, Adaptive Response Rate Exponential Smoothing, Holt-Winters Multiplicative and Holt-Winters Additive models. The authors used the number of cases for the years ranging from 2003 to 2008 to tune the listed systems and the number of cases for the years 2009 and 2010 to evaluate them. Holt's model and Double Exponential Smoothing performed the best and showed a really close error rate. Lin Wu *et al* [19] evaluated the performance of multiple time series forecasting methods. Holt's model performed better than Single Exponential Smoothing. However, traditional ARMA outperformed Holt's model.

Nevertheless, the nature of the data in the related work does not show any similarity to the behavior of load in distributed simulations. The input to the previous systems had either an increasing trend or did not have many oscillations. Thus, previous studies do not show how Holt's model would react towards input streams with data that tends to have a steady trend for a short period of time while oscillating frequently.

The usage of Holt's model in a prediction-based load balancing system for distributed simulations is presented in [3]. By investigating the results and the prediction model, two main weaknesses are identified. First, Holt's model forecasts unrealistic load values when it comes to providing long term load predictions. This is a result of the linear and direct relation between the computed and predicted load and the time intervals applied on the projections. As the time interval increases, Holt tends to provide unrealistic predictions. Secondly, Holt's model does not provide a means to a fast reaction towards sudden or oscillating loads.

Taking into consideration the weaknesses of Holt's model in such systems, different enhancements are presented. These enhancements aim to overcome the previously mentioned drawbacks by applying modifications to Holt's.

III. Predictive Load Balancing Scheme

The modifications and enhancements of Holt's prediction technique are incorporated into the prediction-based dynamic load balancing system described by De Grande and Boukerche [3] which is presented in Figure 1. The load balancing process consists of monitoring the resources in order to define a set of under-loaded and over-loaded resources. Federates are then migrated to suitable shared resources in order to assure a balanced distributed simulation environment.

Each set of shared resources are grouped into a cluster. The cluster master node runs a Cluster Load Balancer (CLB); that handles all shared resources in the cluster along with the federates that run on top of them. CLBs are connected with neighbor CLBs in order to exchange information concerning their own cluster and to do so with Local Load Balancers (LLB) that exist on each shared resource. Moreover, CLBs are connected to a Monitoring Interface and a Prediction Interface. The task of CLBs consists in collecting information about their shared resources and the federates they run. CLBs use the collected information in order to analyze the load and to decide on a set of migration moves. CLBs then initiate the migrations.

The Monitoring Interface provides a means for CLBs to access monitored and related data of resources that are offered

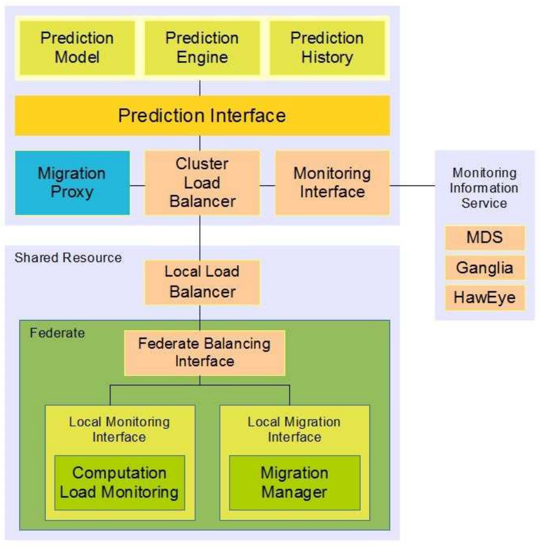


Fig. 1: Prediction-Based Dynamic Load Balancing Architecture

by Monitoring Information Service (MIS). MIS provides each CLB with a set of load related information of concerning under its control. In order to provide such information, MIS uses Grid services. Grid is a system used to manage resources that run distributed applications over shared resources [20]. Access to load information of resources is done through a monitoring and discovery service provided by Grid Index Service.

LLBs are placed on shared resources in order to act as an interface for the resource and federates running on them. LLBs gather the load information of the simulation entity and responses to migration calls. Gathering information is triggered by a CLB, which is forwarded to a Federate Balancing Interface. The Federate Balancing Interface notifies the Local Monitoring Interface for each simulation entity, to retrieve a CPU consumption per federate. A LLB forwards migration calls, which are initiated by CLBs, to the Migration Manager.

Federate Migration is done in two steps, similarly to [21] and [22]. The migration starts with the transferring of static data and initialization files through the Grid Services to the remote resource. Secondly, dynamic execution status data and queued incoming messages are transferred from the local resource to the remote resource by the Migration Manager. Migrations performed between clusters are conducted through the Migration Proxy.

The Prediction Engine, with the assistance of Prediction History and Prediction Model, processes the incoming data. The Prediction History provides a data history which depends on the Prediction Model as each prediction model varies in data history requirements. In this model, a *smoothed value* and a *trend* are saved in the Prediction History as this work emphasizes Holt's double smoothing technique. Prediction Model provides projections based on data provided by the CLB through the Prediction Engine and data from the Prediction History.

A. *Re-distribution Algorithm*

The redistribution algorithm is used to re-arrange the federates in the distributed simulation to have a balanced environment. In order to achieve this goal, the redistribution algorithm of this scheme goes through three different stages: monitoring, re-arrangement, and migration.

The distributed dynamic load balancing system periodically monitors the shared resources. This allows the system to be responsive to load changes. The Monitoring phase is performed periodically every Δt, which is restricted by the periodic refresh rate offered by the monitoring tool. As the projection accuracy depends on the quality of the collected data extracted from the monitoring phase, filtering is performed on the collected data to eliminate values that would affect the prediction. Moreover, the collected data is normalized to enable a fair comparison between the resource loads. Afterwards, the collected data, previous history, and migration status are used to provide load projection in the following three different balancing cycle ranges: short-term, medium-term, and long-term. These projections are used to identify the over-loaded and under-loaded resources in given moments.

With a knowledge of the load of shared resources and the number of projections per balancing cycle, the load balancing system applies a pair matching mechanism between over-loaded and under-loaded resources in each balancing cycle. The load balancing system assigns a greater importance to projections that are closer to the current time. Before the system applies the pair matching mechanism, the system first sorts the list of resources from the most over-loaded to the most under-loaded. The pair matching mechanism is performed on the ordered list.

The mechanism initiates a migration call if the difference between the load of the over-loaded and the under-loaded resources is beyond a threshold. Based on the local migrations and their success rate, an inter-domain migration process is started. Inter-domain migration is initiated by a CLB requesting a Cluster Load from its neighbor CLBs. The Cluster Load of a CLB is the average of the loads of its resources. Once the Cluster Load is received by the initiator, the initiator will perform a selection mechanism to identify load imbalances between resources. Resources that present loads that exceed a threshold are considered as candidates for the remote redistribution mechanism functioning between domains. Once a neighbor CLB receives the list of candidates, it performs an inter-domain redistribution mechanism on the three balancing cycles by comparing remote over-loaded resources and local under-loaded resources. Similar to the process of local balancing, a migration call is generated once the difference in load between the two resources justifies a load imbalance by exceeding a threshold. At the end of the selection process, all migration calls are sent at once to the requester CLB.

B. *Forecasting Load Status*

The distributed load balancing system performs on three different load forecasting levels: short, medium, and long term. The three projections are independent from the prediction model. Each projection is designed to provide a goal to the system. Short term projection (SP) aims to solve current load imbalances while medium and long term projections (MP and

LP) are used to prevent load imbalances that might happen. Each projection represents the prediction of load in certain balancing cycles. Short, medium, and long term cycles are then defined as 1, 3, and 5 balancing cycles, respectively.

The threshold used to compare the over-loaded and under-loaded resources is adaptive; and it is modified based on the direction of the tendency of a resource load. The amount of adjustment needed is proportional to the balancing cycle. Thus, medium and long term projections receive larger adjustments.

C. Prediction Models

The computational load of each shared resource is collected in a list recorded in fixed time 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.

EWMA is a well-known time series prediction technique that has been used in different studies. This technique observes the internal relationship of elements in three different aspects [23]. Single smoothed EWMA is an exponentially averaging technique that computes the smoothed value of the predicted term. Double EWMA adds a trend to the exponentially calculated average. The trend of predicting the load of shared resources is used to identify the tendency of the resource load. Triple EWMA requires a deep understanding of the system in order to detect the seasonality in the time-series data, along with the trend, and to use this knowledge to properly predict the load. Throughout the process of tuning and evaluating the performance of the proposed variants, the distributed simulation showed no sign of seasonality in its time series. Thus, the Triple smoothed EWMA is not a suitable method for forecasting load of HLA-Based distributed simulations.

Holt's model, understood as a time series prediction technique represented by the formulas 1 and 2, is applied to the collected data in order to forecast the load. This is done by computing the value of F_{m+i} in Formula 3. The double exponentially smoothing computes the predicted load on the collected list of loads. It first finds the current smoothed value, sum_i, based on the current actual load: $elem_i$. In addition, it uses the previous smoothed value, sum_{i-1}, and the previous trend, t_{i-1}, in order to add the smoothness factor. The trend enables the extrapolation of the average of the smoothed value; and, its calculation is based on the tendency of the smoothed value, $sum_i - sum_{i-1}$, and on the previous trend: t_{i-1}. The tendency of the load is defined by the sign of the trend, where a positive sign presents an increasing tendency and a negative sign shows a decreasing tendency. Formula 3 is used to forecast a future value for a given time interval m. Forecasting a load using Holt's model requires setting up two constants: α, the data smoothing factor, and β, the trend smoothing factor.

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

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

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

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

In terms of initializing Holt's model, sum_0 requires a knowledge of the sum_{-1} which is not set in the beginning of the simulation. Thus, as a precautionary step, sum_0 is set to the $elem_0$ as there exists no previous data that sum_0 can make reference to. As in the case for the trend, t_i requires a knowledge of the previous trend. When the system is initialized, t_0 is set to 0, because it is not known whether the system is increasing or decreasing. Thus, t_1 needs to be computed as $t_1 = elem_1 - elem_0$ to represent the actual trend that Holt's model can use for the following predictions.

With the implementation of Holt's model in such dynamic balancing systems, two main problems were identified. The first problem resides in the direct linear relation between the time interval, m, which represents the number of balancing cycles needed to calculate SP, MP, and LP, and the value of the prediction as represented in Formula 3. The system uses three different time intervals to calculate the three different projections (SP, MP, and LP), in order to prevent load imbalances in the distributed simulation; thus, the effect of this problem appears as the time interval increases. Figure 2 represents a real load of a shared resource and the three different projections calculated by the dynamic load balancing system. The first problem becomes clear when the load suddenly oscillates. Such oscillations, especially if they continue for a number of cycles, will cause the value of the computed trend to increase or decrease dramatically. These changes in the trend are multiplied by the time interval for each projection and added to the smoothed value. The multiplication component will add a significant value to the computed projection and it increases as the time interval increases.

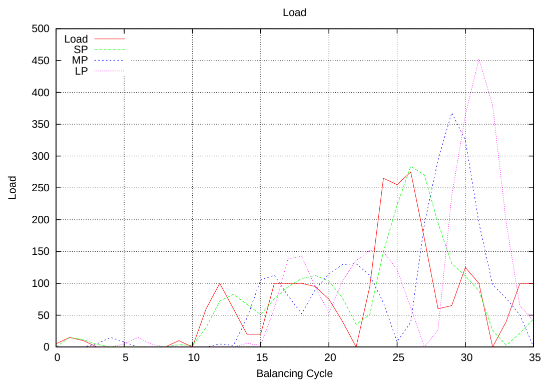


Fig. 2: Representation of the first problem

The second problem takes the form of slow responses to sudden load oscillations. This is a result of the direct relation between t_i and t_{i-1} in Formula 2. When a simulation entity is migrated from, or to, a shared resource, the load of the resource oscillates to reflect the action of migration. The impact of these sudden oscillations on load trends sent to t_i is affected by the trend smoothing factor: β. Figure 3 represents a real data sample of a distributed simulation and the short-term projection calculated by Holt's model. By observing the data sample in Figure 3, it can be understood that the nature of the data does not follow a certain tendency for more than 5 balancing cycles. Once the actual trend in the data sample oscillates, it takes the calculated sum_i a number of balancing cycles to become adjusted through the application of Holt's model. The purpose of this adjustment is to keep in sync with the real load.

1) Cutoff: Cutoff is a naive approach aimed at solving the first problem listed. Cutoff defines a threshold that represents the highest load the system has come across. The threshold is used to limit the predicted load from exceeding what the balancing system has detected. Once a projection is calculated, the projection is compared to the threshold. The minimum value is selected in order to assure a limitation on projections.

$$SP = min(F_{m+1}, threshold) \tag{5}$$

$$MP = min(F_{m+3}, threshold) \tag{6}$$

$$LP = min(F_{m+5}, threshold) \tag{7}$$

At the initialization time, the $threshold$ is set to $elem_0$. The predicted values of different time intervals are calculated and limited by a threshold. The threshold is compared against $elem_i$ during every balancing cycle and is set to the maximum of both.

With the implementation of the threshold, the Cutoff is expected to decrease the average error rate generated for medium and long-term predictions; this is the case because this technique attempts to prevent unrealistic projections, especially when the load oscillates, by limiting projections using the threshold. Cutoff, however, might limit the ability of Holt's model to predict short-term projections as it will create a hidden relationship between the current projection and the threshold. This behavior might decrease the accuracy of the projections for short-term projections.

The degradation experienced in calculating the short-term projections would increase performance problems when the variant is implemented on the dynamic load balancing system. As the current dynamic load balancing system assigns more importance to short-term predictions, when generating the migration list, the incorrect computations for the short-term projections would result in an imbalanced distributed environment.

2) Reversed Trend (RT): Reversed-Trend is a proposed approach aiming to make the dynamic load balancing system respond to load oscillations faster. A deep analysis of the behavior of the trend shows that the trend responds slowly to load oscillations. This is a dependency problem, as the t_i depends on t_{i-1}. By looking at Figure 3, it can be noted that the predicted load computed by Holt's model does not exactly follow the real load when there are sudden load oscillations.

Upon looking at the trend calculated by Holt's model for the prediction of the load, it was observed that sign of the the trend does not change immediately in a way that reflects the changes in the actual tendency. The sign of the trend is important as it represents the tendency of the load. Moreover, it is used to predict the load in future time intervals. Thus, reverse-trend changes the sign of the trend to match the actual tendency.

This approach compares the computed tendency of Holt's model, t_i, at the current load with the real tendency $elem_i - elem_{i-1}$. If the directions of the tendency do not match, the reverse-trend technique will *reverse* the sign of the computed trend, that is used to compute the smoothed value; this will be done to match the real tendency. Before the inversion, the Reverse Trend first calculates the differences in load. The reason for this calculation is to prevent *unnecessary sign reversion* in small load oscillations.

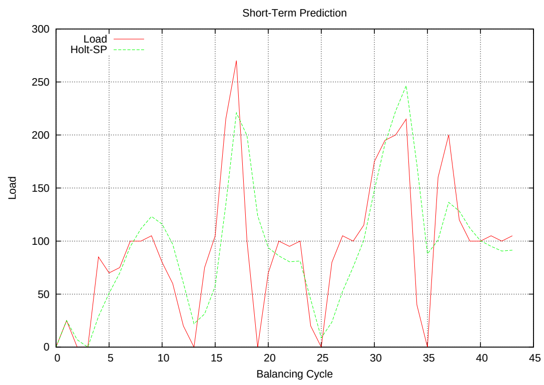


Fig. 3: Representation of the second problem

As Reverse Trend adopts more sensitivity while reacting towards load oscillations and sharpness in projections, Reverse Trend is expected to provide better readings for short and medium-term projections. However, the performance achieved while projecting long-term loads might decrease; this is because the long-term projections are needed to be as smooth as possible and not sensitive to load oscillations.

3) Separate Systems (SS): Separate Systems is an approach that aims to solve the first problem described in Holt's model. Holt's model uses a fixed α and β in order to calculate the prediction of each time interval. α represents the percentage of effect the previous smoothing values have on the current smoothing value. From formula 1, the larger α is, the greater the impact is of $elem_i$ on the smoothed value and the smaller impact of the previous smoothing value is on the current smoothed value.

From observations, short term projection depends mostly on $elem_i$. On the other hand, medium and long term predictions depend more on previous smoothing values: sum_{i-1}. Thus, different fixed constants for each projection of SP, MP, and LP are used to apply emphasis on each type of prediction. This results in having three different systems, where each system is responsible for calculating a projection. The short term prediction system would have a larger value of α to exercise more influence of $elem_i$ to sum_i. Unlike the short term prediction system, the medium term prediction system presents a smaller α to emphasis sum_{i-1}. The long term prediction system then is set to present a smaller α than the medium term prediction system.

The Separate Systems is expected to provide better performance for all projections. This is a result of providing a different set of α and β parameters for each system that reflects the importance of the previously smoothed value and trends for each range of forecasting.

IV. Experimental Results

In the evaluation of the proposed approaches, the enhancements are compared with the prediction-based distributed load balancing system [3]. The evaluation process is done in two stages. The first stage performs the evaluation of the prediction techniques on data samples, generated by running several distributed simulations on top of a set of shared resources. Moreover, a systematic computational load was used in the system to emulate simulation overhead. The aim of the first stage is to test the efficiency of the methods and to properly configure them for real simulations. The second stage is to implement the enhancements on the distributed load balancing system. The second stage aims to verify the conclusions of the first stage on the subject of real distributed simulations. A basic implementation is conducted on a dynamic load balancing system as shown in Figures 9a and 9b.

1) First Stage: Different data samples were collected and analyzed in order to have test cases that cover the possible scenarios distributed simulations might experience. These data samples were classified into the following 3 different categories based on the number of federates: low, medium, high. In each category, different cases were taken into account, such as actual loads that follow a specific pattern. The goal is to cover as many situations as possible in order to test the close-to-real performance of the different enhancements instead of testing the systems against a specific type of situation. Figure 4 shows a snapshot of three data samples of different load oscillation frequencies.

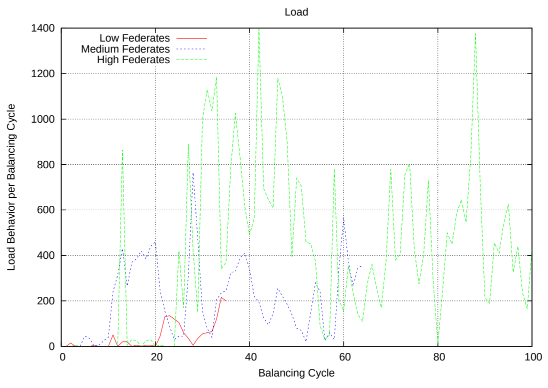


Fig. 4: Representation of data samples with different setups

Because of the importance of α and β in the context of system performance, a grid search was performed where a set of possible values for α and β, $0 \leq \alpha, \beta \leq 1$, with a step of 0.01, were thoroughly compared and the combination with the smaller amount of computed error was chosen. The predictions were extensively evaluated and compared using the calculated error by the enhancement. The error may be understood as the difference between the predicted load and the real load value. The average for the errors of each category and each projection was determined, and this is shown in the following tables.

For the unmodified Holt's model, the grid search returns $\alpha = 0.36$ and $\beta = 0.36$. Table I lists the rounded average error and the standard deviation, in brackets, of the unmodified Holt's model on the data samples for each projection.

A set of experiments were conducted to compare the performance of Holt's model and Cutoff. The naive approach (Cutoff) does limit the medium and long term projections as it is supposed to do. Figure 5 demonstrates the limiting effect on the long-term prediction. As a result of this limitation on the projections, the Cutoff has set the projections to a threshold instead of to unrealistic projections. This results in a decrease in the error rate of medium and long projections, as shown in Table I. However, the main task of designing the Cutoff has a negative side-affect on the short term projections. This method decreased the performance of short term projections as a result of applying measures instead of enabling a benefiting from the capabilities of Holt's limiting model.

TABLE I: Average Error and STD for the different variants

	Holt's	Cutoff	RT	SS	SSRT
SP	86(15)	88(15)	55(11)	31(5)	25(4)
MP	193(32)	185(31)	181(30)	166(25)	139(24)
LP	217(41)	199(36)	205(37)	149(25)	141(24)

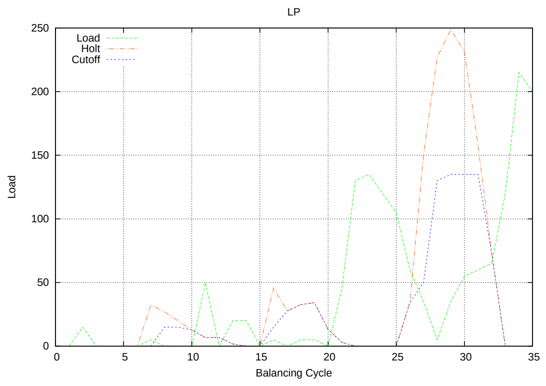


Fig. 5: Comparison between Holt's model and Cutoff

Reversed-Trend was implemented and evaluated against the data samples. The results, as shown in Table I, show that the Reversed-Trend improves the response speed to load oscillations for short and medium-term projections when compared against Holt's model. Figure 6 demonstrates the fast response of RT to oscillations against Holt's model for short-term projections. However, looking at the performance for the long-term prediction of Reversed-Trend and Cutoff, the result of *sharpness* added by the Reverse-Trend, affects the precision of long-term predictions.

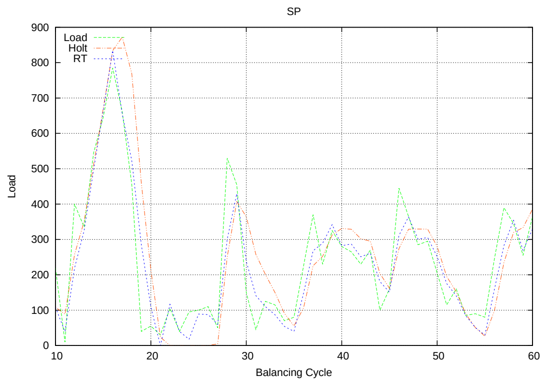


Fig. 6: Comparison between Holt's model and RT

By breaking down the long-term predictions for the different categories, the *sharpness* effect adds more accuracy to the predicted value in the case where the load oscillates more frequently; this happens when the frequency of federates is high. This situation is presented by Table II as the Reverse

TABLE II: Breakdown of LP in Cutoff and RT

Frequency	Cutoff- Avg Err	Cutoff-STD	RT-Avg Err	RT-STD
Low	65.67	29.64	71.70	32.21
Medium	180.15	42.97	198.43	42.02
High	351.97	36.27	343.58	38.23

Trend provides less error for long-term projections with a high number of migrations.

A grid search was performed on each system in the Separate System technique. As expected, a low α and β are needed for MP and LP as the dependency on previous loads becomes more important. By calculating the errors, $(\alpha = 0.85, \beta = 0.36)$ returns the smaller amount of errors for SP. Similarly, $(\alpha = 0.08, \beta = 0.09)$ and $(\alpha = 0.05, \beta = 0.05)$ return the smaller amount of errors for MP and LP, consecutively. Table I shows that by having a separate system for each projection, a higher degree of prediction precision is possible than when calculating the projections with Holt's model system and different time intervals. Figure 7 demonstrates the higher amount of precision the Separate System technique offers compared to Holt's model.

Another variant comes to mind after exploring the different proposed solutions where *each* solve *one* of the previously listed problems. This variant combines the ability of two systems to have a combination that solves both the observed problems with Holt's model. By merging the *Separate Systems* technique with the *Reversed Trend* technique, and by thus generating a *Seperate System Reversed Trend* (SSRT), the performance is expected to improve. RT will add an elemnt of sharpness and quick reaction to sudden oscillations. Moreover, SS would increase the precision of MP and LP. As expected, Table I shows that the accuracy of the new variant improved for all projections.

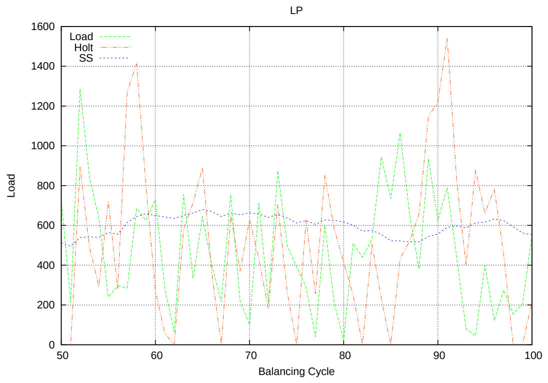


Fig. 7: Comparison between Holt's model and SS

Figure 8 summarizes the results of the different Holt variants. Each bar represents the average number of errors for an enhancement with the confidence rate at 95% for each projection. Cutoff shows that it is capable of decreasing the error of projections with the limitations it sets to the predicted values. However, the limitations imposed on short term predictions add restrictions to the Holt's model performance which reduces the performance of the short term predictions. On the other hand, RT and its additional sharpness to the forecast affects the performance of long-term predictions. Observing the implementation of RT on different data samples and comparing the results with other models, raises the need to have smooth long term predictions instead of a sharp prediction for low and medium number of federates. However, the sharpness does add value to the long-term prediction when the number of federates is high and the load of the shared resources oscillate more frequently. The isolated system in SS improved the predictions as it gives each projection a different treatment for the two factors of Holt's model. These different factors show that implementing only one Holt's model is not an ideal solution for data that oscillates frequently. Combining Separate Systems with Reverse Trend shows that the two variants together tackle the drawbacks listed and decrease the total number of error in the projections.

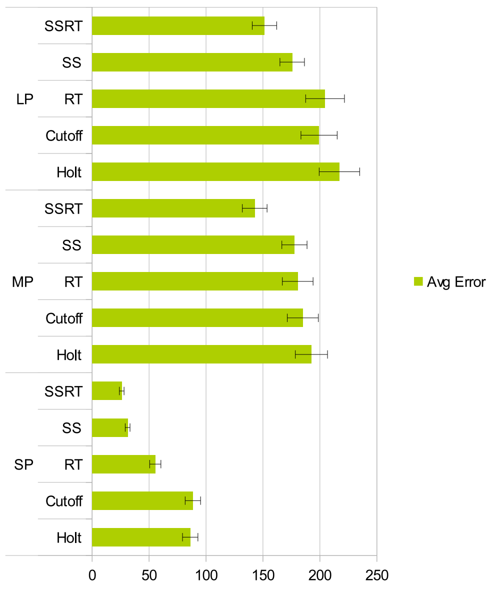
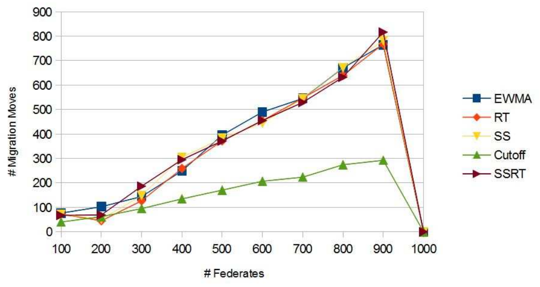


Fig. 8: Avg Error and STD for variant

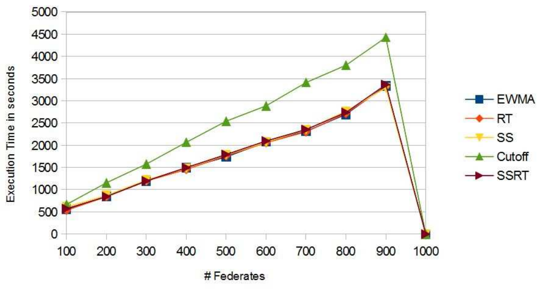
2) Second Stage: The testbed for the experiments was a heterogeneous environment that consisted of federates deployed in the following two clusters of computing servers: IBM and Dell. The IBM cluster was composed of 23 computing servers interconnected by a gigabit Ethernet network; each server contains a Core 2 Duo 3.4 GHz Intel Xeon CPU and 2GB of DIMM DDR RAM. The Dell cluster contains 17 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 two clusters are interconnected through their management nodes with a Fast-Ethernet link. Linux operating system was installed in every server. Globus Toolkit 4.2.1 was set up to support the balancing systems; and, HLA platform with RTI version 1.3 was used to coordinate the experimental simulations.

For the balancing systems were evaluated in this experiment, 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 deployed federates on the 40 shared resources and one of the servers was dedicated to running the HLA RTI executive. In the simulation scenario, 1 to 1000 federates controlled the movement of objects (tanks) during 100 time steps. Limited to the communication influence on the simulations, each federate managed only one object and calculated the movement of its tank through highly intensive computational tasks.

In this initial analysis of our proposed solutions with real simulations, Holt's model and the proposed variants demonstrate similar performance for almost all of the different models. Threshold restrictions applied by Cutoff have ignored some of the migrations that should have been generated by the dynamic load balancing system to balance the shared resources as shown in Figure 9a. As a result, Cutoff yields an unbalanced simulation environment with over-loaded shared resources; therefore, this leads to an increase in the execution time, as shown in Figure 9b. Unlike Cutoff, the other variants provide the same performance as Holt's model. However, analysis and examination are being conducted to understand, in depth, the reasons behind the similarities in performance.



(a) Number of migrations performed per variant



(b) Execution time per variant

Fig. 9: Performance on Real-Time Simulation

V. Conclusion

In this paper, a number of Holt's variants are proposed for load redistribution on a Large-scale HLA-Based distributed simulation. The enhancements are the result of identifying two weaknesses in Holt's model implementation on a prediction-based dynamic load balancing system. Cutoff and Separate Systems solve the problem of the direct linear relationship between the time interval and the projection. Reverse Trend solves the other identified weakness, by providing a faster response to sudden load oscillations. The enhancements reacted as expected. Separate Systems and Reverse Trend were merged in order to create an enhancement that deals with both weaknesses. The performance of the merged proposal results in better projections. The initial implementation of the different enhancements did not return results that reflect the expected improvements on the dynamic load balancing system.

For future work, an in depth analysis is required to modify the dynamic load balancing system to accommodate the proposed variants. Also, more analysis of Holt's model is needed to identify some weaknesses of the implementation in similar systems. Moreover, different prediction techniques will be implemented in the dynamic load balancing system in order to compare the gain in performance in the case of detecting load imbalances and reacting towards load changes. In addition, dynamic adjustments of α and β are needed to achieve a dynamic reaction towards load oscillations on shared resources.

References

[1] R. De Grande and A. Boukerche, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," in *ISCC*.

[2] ——, "Dynamic load redistribution based on migration latency analysis for distributed virtual simulations," in *HAVE*.

[3] ——, "Predictive dynamic load balancing for large-scale hla-based simulations," in *DS-RT*.

[4] R. Schlagenhaft, M. Ruhwandl, C. Sporrer, and H. Bauer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," in *PADS*.

[5] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," in *PADS*.

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

[7] P. Peschlow, T. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *PADS*.

[8] D. Glazer and C. Tropper, "On process migration and load balancing in time warp," *IEEE Trans. on Parallel and Distributed Systems*.

[9] C. Burdorf and J. Marti, "Load balancing strategies for time warp on multi-user workstations," *The Computer Journal*.

[10] C. Carothers and R. M. Fujimoto, "Background execution of time warp programs," in *PADS*.

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

[12] E. Deelman and B. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems," in *PADS*.

[13] A. Boukerche and S. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *PADS*.

[14] A. Boukerche and R. De Grande, "Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems," in *DS-RT*.

[15] R. E. De Grande and A. Boukerche, "A dynamic, distributed, hierarchical load balancing for hla-based simulations on large-scale environments," in *Proc. of the 16th international Euro-Par conference on Parallel processing*.

[16] L. Xia, G. Yaomei, and S. Weiwei, "Forecast to textile and garment exports based on holt model," in *ISME*.

[17] S. Abdullah, N. Sapii, S. Dir, and T. M. T. Jalal, "Application of univariate forecasting models of tuberculosis cases in kelantan," in *ICSSBE*.

[18] A. Padyana, C. Sudheer, P. Baruah, and A. Srinivasan, "High throughput compression of floating point numbers on graphical processing units," in *PDGC*.

[19] L. Wu, J. Yan, and Y. Fan, "Data mining algorithms and statistical analysis for sales data forecast," in *CSO*.

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

[21] A. Boukerche and R. Grande, "Optimized federate migration for large-scale hla-based simulations," in *DS-RT*.

[22] Z. Li, W. Cai, S. Turner, and K. Pan, "Federate migration in a service oriented hla rti," in *DS-RT*.

[23] E. S. G. Jr., "Exponential smoothing: The state of the art - part ii," *Int. Journal of Forecasting*.