DTHMM ExaLB: discrete-time hidden Markov model for load balancing in distributed exascale computing environment

Ulphat Bakhishoff¹, Ehsan Mousavi Khaneghah²*, Araz R. Aliev¹ and Amirhossein Reyhani Showkatabadi³

Abstract: In high performance computing (HPC) systems, the load-balancing manager decides on performing reloading related activities based on information received on the system state. In Distributed Exascale systems, unlike traditional HPC systems, dynamic and interactive events occur in a system that changes system status, therefore, they violate the activities associated with the load-balancing manager. Managing these types of events, require a thorough analysis of the dynamic and interactive occurrences that lead to this situation. In this article, dynamic and interactive events, which violate the function and activity of the burden distribution manager based on the Discrete-Time Hidden Markov Models, were analyzed. This mathematical model was used to analyze dynamic and Interactive events on the system state. The mathematical model presented in this article provides such ability for the load balancer that instead of analyzing each dynamic and interactive event, do the activity based on changes, which violates functionality of load balancer, in system state. Based on this model, load balancer manages to reload the system when dynamic and interactive events occur. This makes the load balancer, which is able to continue the process of implementing its previous activities, taking into account not only dynamic and interactive behaviors, but also the new state of the system. As the result of evolutions it is got that the model makes an opportunity to predict load difference on resources based on simple mathematical calculations while dynamic and interactive event is occurred.

ABOUT THE AUTHOR
Ulphat Bakhishoff is a Ph.D. student of General and Applied Mathematics Department of Azerbaijan State Oil and Industry University. He received a master's degree after attending Azerbaijan Technical University on specialty Software Development. He took a bachelor's degree in Azerbaijan State Oil and Industry University on specialty Applied Mathematics. His research interest is the design and development of load balancing models for distributed computing systems. His current research is on the mathematical model of load balancing for distributed Exascale computing systems.

PUBLIC INTEREST STATEMENT
One of the big challenges for Exascale systems is keeping the system is reloaded after Dynamic and Interactive events occurred. There are two types of solutions: First, allowing dynamic and interactive events to occur and then reloading the system; second, predicting the influences of dynamic and interactive events on the system and reloading the system while dynamic and interactive events occurring. In this article, a Discrete-Time Hidden Markov Models (DT HMM)-based mathematical model is proposed for Load Balancing to be able to predict the influences of Dynamic and Interactive events on the system. This model makes an opportunity to predict load difference on resources based on simple mathematical calculations while the dynamic and interactive event has occurred.
1. Introduction

The mission of Load Balancer is managing activity to obtain performance in system. Load Balancer should load a system based on resource-attribute and process-requirement in such manner, that any process responds in minimal time and none of any available resources remain unused (Ehsan Mousavi Khaneghah et al., 2018; Mirtaheri & Grandinetti, 2017; Xiao et al., 2017). For this purpose, load balancer collects a set of information about current resource status and a set of information about process requirements. Load balancer have to be able to do three works: collecting information form processes and system according to some benchmarks; making decision on a set of policies; using set of tools for example, process migration or resource discovery to perform policies. All these are possible when the system surrounding load balancer does not change, while load balancing in action (Patil & Gopal, 2017; Van der Wijngaart et al., 2017).

Information, which collected from system by load balancer, must be valid until load balancer finishes its work. The information known to load balancer and real information about real state of the system must be same in any time while load balancer works (Soltani et al., 2012; Thakur & Kumar, 2015).

In cluster-computing environment (Sharifi et al., 2008), complete sets of process requirements and resource attributes are predetermined. This type of HPC system is designed and configured based on these process requirements and resource attributes and then starts working. That is why, the system does not change at runtime and the condition defined above is applicable in such systems. In this case, load-balancer starts at startup of system before processes started (Sharifi et al., 2008; Sharma & Aggarwal, 2015; Thoman et al., 2018).

In grid computing environment, process requirements are predefined but resource attributes are dynamic. Resources can be increased or decreased dynamically at runtime. Such systems are designed and configured using initial process requirements and resource attributes (Jairam et al., 2015; Khan et al., 2017). In addition, each time resource attributes changed, Load Balancer starts working. In this type of HPCs, changes are not intensive. For this reason, it is possible to find a situation that system and process status is stable while load balancing works (Meddeber & Yagoubi, 2016; Raghu et al., 2018; Rajkumar et al., 2015).

However, in Distributed Exascale computing environments (Mirtaheri et al., 2013; Reed & Dongarra, 2015) may occur Dynamic and Interactive event (Ehsan Mousavi Khaneghah, ShowkatAbad, Ghahroodi et al., 2018; Inghs et al., 2015). Both process requirements and resource attributes are dynamic (Ehsan Mousavi Khaneghah et al., 2018; Ehsan Mousavi Khaneghah, ShowkatAbad, Ghahroodi et al., 2018). Resource attribute dynamicity is more intensive than grid systems (Raghu et al., 2018; Rajkumar et al., 2015). Process requirements dynamicity could be in the one of following three forms (Alowayyed et al., 2017; Ehsan Mousavi Khaneghah, ShowkatAbad, Shadnoush et al., 2018). First, process forks new process or processes, second, process communicates with another process, third, process interacts with system environments. Dynamic and Interactive events may cause occurrence state which not defined initial response structure (Ehsan Mousavi Khaneghah et al., 2018; Ehsan Mousavi Khaneghah, ShowkatAbad, Ghahroodi et al., 2018; Ehsan Mousavi Khaneghah, ShowkatAbad, Shadnoush et al., 2018). The most important influence of Dynamic and Interactive events within Load Balancing is that information gathered by Load Balancing maybe turn to invalid (Ehsan Mousavi Khaneghah et al., 2018). Load Balancer has three choices when Dynamic and Interactive events occurred. One of these is that, Load Balancing is activated as a result of a system state change event or time occurrence.
Any state changes of system are ignored until the end of activity. At the end of activity, it is checked whether if the current system status is as same as calculated or not. If the answer is no, restarts load distribution activity. The next choice is that, Load Balancer observes all Dynamic and Interactive events, analyses them and makes policies for them. For supporting this, all Dynamic and Interactive events and their influence within Load Balancing have to be investigated. It is hard work. Because of this, a pattern for all dynamic and interactive events and the way of their influence on Load Balancer is requires. The last choice is defining what Dynamic and Interactive event result can change system status, which affects decision of Load Balancer (Ehsan Mousavi Khaneghah et al., 2018).

This is easy solution, but its shortcoming is that, result of work of Load Balancing may be different from reality. Because, it works on model which created based on information gathered from processes and system. This information may be valid or invalid. According to information collected from different agents, a single node may at the same time marked as both available and failed because of limited communication between nodes (Shang, 2018). It may result Load Balancer to restart at the end of its work. It conflicted with HPC principles. Because HPC tries to finish its works in some response times, but in this case system may do some repeated operations.

The model described in this article, without analyzing the concept of dynamic and interactive nature of processes and also the initial state of the load distribution and just by considering the result of the dynamic and interactive event, predict the load difference on resources based on Discrete-Time Hidden Markov Model (DT HMM) and let the Load Balancer decide with more precision.

## 2. Related works

Traditional HPC systems are designed and developed based on static Problem Technical Accounting (PTA). Due to this, the load distribution management complies with Eq. (1) (Ehsan Mousavi Khaneghah et al., 2018; Ehsan Mousavi Khaneghah, ShowkatAbad, Ghahroodi et al., 2018)

$$\text{LoadDistribution} : [\text{ProcessRequirements}] \rightarrow [\text{Resourcespace}]$$

Therefore

$$\text{BestLoadDistribution} = \left[ 0 = \frac{\text{Process} \in \text{HPCprocess}}{\text{Pairwise} \Rightarrow \text{HPCProcessScheduling}} \right]$$

$$\text{and}$$

$$\text{ResourceActivity} = 100\%$$

However, in Distributed Exascale Computing Environment, dynamic and interactive event may occur anytime, even while load distribution works. That’s why, Eq. (1) is not applicable in Distributed Exascale computing environments. One of the works on Load Balancing in Distributed Exascale Computing Environment is AMRC (Khaneghah & Mohsen, 2014). This work covers the early stage of Load Balancing. According to the article, Load Balancing is an allocation process. There are two types of it. First, restoring the system to its initial state, and Second, bringing the system to a new stable state. It is proposed restoring the system to its initial state in AMRC. In this work used vector algebra to create model. In the model, two vectors defined to solve the problem—the vector of the stable state of the system and the vector of the current state of the system. If these two vectors are not the same, the system is unstable and has to be stabilized. Difference between these two vectors may be existence a non-zero angle between them or difference between their values. To stabilizing, the system it has to be removed these differences. One shortcoming of this model is that creating defined vectors is time-consuming. Another shortcoming is it is not possible always restoring the system to its initial state.

However, in (Sharifi et al., 2010) it is proposed bringing the system to a new stable state (Khaneghah, 2017). They proposed load-balancing model for four types of resources—File, Memory, I/O, and CPU. The model based on supply and demand model. Therefore, depending on
request pattern each node in the P2P system can be both client and server at the same time for this model. Doing this, each machine of the system has to be informed about all other machines. Gathering this information takes a great deal of time.

According to the model proposed in (Mirtaheri & Grandinetti, 2017), machines in the system are divided into two groups, machines with positive load and machines with negative load. Machines with positive load are machines that getting loaded. In addition, machines with negative load are being unloaded. According to the model for load balancing, loads have to move from positive loaded machines to negative loaded machines. Shortcoming of the model is it is one-dimensional. So, saying load it is intended only process.

3. DTHMM ExaLB

In traditional high performance computing (HPC) systems such as cluster, Grid and P2P computing systems, each computing node will be described with only CPU usage, from the point of load balancer view (Elfaham et al., 2016; Jiang, 2016). An enhanced model that is described by three parameters <CPU usage, Bandwidth, Memory usage> is also valid but not frequently used.

Unlike traditional HPCs, in Distributed Exascale computing environment, dynamic and interactive event may occur like in High Energy Physics, Climate Simulation, Brain Simulation, and Space Weather (https://www.deep-projects.eu/applications/project-applications.html). Three types of dynamic and interactive events may occur in Exascale system: Process can create new process and increases the count of running parallel processes. Process may communicate with another process, so, multiple processes may use shared artifacts. In this case, these processes can be executed as single process, which has sequential operations in it. So, this event decreases count of running parallel process. The third event is that process interacts with environment. In this case the count of running processes doesn’t change. But it changes the requirements of the process. All these events create new process requirements for the resource attributes (Khaneghah & Mohsen, 2014). It is the global approach using Eq. (1) in all systems, but in Distributed Exascale Computing System will not be able to answer all kinds of limitation and constraints about load distribution. Because, dynamic and interactive events are not considered in this formula. So that, while load distribution continues, dynamic and interactive events may occur (Ehsan Mousavi Khaneghah et al., 2018; Khaneghah, 2017). This leads to change in process requirements, resource attributes, and load distribution did not consider this. In this case the current work of Load Balancer becomes useless. By considering the concept of global activity and dynamic and interactive nature of processes, <Process state, Global state, Request state> can be a description of each computing node from the load balancer’s point of view.

After considering the dynamic and interactive nature of processes in Eq. (1), the following Equation will be accessed.

\[
\text{LoadDistribution(DynamicInteractiveEvent)} : \quad \text{Process}_{\text{requirement}} | \text{Global Activity}_{\text{condition}} \rightarrow \text{Resource}_{\text{space}}
\]

As Eq. (2) shown, process requirements are closely under the control of the global activity, by occurring a dynamic and interactive event the nature of the process and the role of it in the global activity, which is joined, may change. Eq. (2) by considering the constraints and the limitation of the process and the possibility of occurring a dynamic and interactive event tries to map process dependencies to the resource space.

It is important considering that Eq. (2) is abstract pattern. However, it considers dynamic and interactive events, it is impossible to implement it, because for implementing one-time or maybe more than one execution of this activity for understanding the initiations, data requirements, and decelerations is needed. Because, in this case, a creation an implementation, which considers each
Each dynamic and interactive event makes change on system state.

\[ \text{DynamicInteractiveEvent} : \text{SystemState} \rightarrow \text{SystemNewState} \] (3)

By Considering Eq. (3), formulation of load distribution based on system state and dynamic and interactive events is possible. Instead of considering whole of the system, the system will be observed at the \([t_1, t_1 + \epsilon]\). Unlike cluster computing systems, the local load balancer will check the load of the system. The load balancer instead of proposing each dynamic and interactive event begins to intend each new system states. Now, the load balancer has the ability of redistribute the load of the system, the Eq. (4) describes a load balancer, which is able to cope with dynamic and interactive events.

\[ \text{LoadDistribution} (\text{SystemState}) : \text{(Process requirement | Global Activity Condition)} \rightarrow \text{ResourceSpace} \] (4)

Eq. (4) gives us an opportunity making load distribution without considering dynamic and interactive events.

At the beginning, HPC expert designs and develops HPC system based on initial PTA, executes process and leaves system and system enters execution phase (Khaneghah & Mohsen, 2014). All process requirements and all demanded initiations are gathered in initial PTA so, the first stable state for system, will be the initial PTA. Because the system is configured using static process-requirements and static resources. In this article, saying resource considered four types—File, Memory, I/O, and CPU (Khaneghah, 2017). If any dynamic and interactive event occurred, the system enters to the new state (Khaneghah, 2017). For the proposed approach, this status has to be appended to the state set of system and load distribution has to be restarted. Current parameters of the system are chosen as the initial parameters of the new state. All resource request history is used for the optimization of load distribution. If the system state changed to another known state as the result of dynamic and interactive event, it means state set reach to stability. In this case, it can be chosen best state configuration for current resource request. This is the new stable state for system.

After clarifying the whole problem, it can be stated that a model without any bounding on transmission states and the result of transaction is needed. The possible models that can be used for modeling the mentioned problems can be data flow modeling (DFM) (Li et al., 2015; Valacich et al., 2017; 2019), automata theory (Kupferman, 2018; Najim & Poznyak, 2014) and Hidden Markov Model (HMM) (Vangelov & Barahona, 2017; Zucchini et al., 2016). By analyzing the implementation of each model, the Hidden Markov Model is chosen. This model is used to model processes of systems like dialogue systems (Chung-Hsien et al., 1998), conversational recommender systems (Gaeta et al., 2017) that have changed over time and are constantly improving based on current observations. In this article, it applied to load distribution while dynamic and interactive events occur.

3.1. DTHMM a load distribution model

In article proposed applying DTHMM (Vangelov & Barahona, 2017; Zucchini et al., 2016) to the solution of problem. This model consists of following elements:

\[ S = \{s_1, s_2, \ldots, s_N\} \]— set of known states of the system. \(N\)— State count.

\[ V = \{v_1, v_2, \ldots, v_K\} \]— set of nodes which distributed in system. \(K\)— object count.

\[ O = \{o_1, o_2, \ldots, o_T\} \]— set of requests given at time \(t\). \(T\)— is an observation count. At the same time, system gets more than one parallel request. That is why each observation element in at time moment has to contain requests at the same time. Considering this, we use two-dimensional
observation models. Where \( t \) indicates observation time, and \( r \) indicates resource attribute type. So \( o_t^r \) is the given request to the resource \( r \) at time \( t \).

\[
O = \left\{ \left\{ o_t^r \right\} \right\}, 1 \leq t \leq T, 1 \leq r \leq R
\]

\[
Q = \{ q_1, q_2, \ldots, q_T \} \text{— set of states which are passed while observation.}
\]

The first concept, which should be declared, is the concept of state. In this article, the concept of states is discussed as all situations of computing nodes, in which the load of the computing system is calculated for. The calculation trend is influenced by the nature of the system. In traditional computing systems such as clusters, the \(<\text{CPU utilization}, \text{bandwidth}, \text{memory usage}>\) can be defined as the indicators of load distribution. In distributed Exascale computing systems, the load of the computing system, can be defined and described with the concept of global activity.

Four sets as indicators of the DTHMM is used. The indicators are stated in a way, which are completely cover all requirements and meet all constraints of the system. The first set which can be the basis of modeling is the sets, that contains all stable states that are seen during the \([\alpha, \beta]\), where \( \alpha \) and \( \beta \) are two stable points of the system. Another set for gathering the object of load distribution is set \( V \). Objects which are gathered in set \( V \), are all computing nodes which the Load Balancer begins to reload Balance their load in \([\alpha, \beta]\). As explained above the observations and the number of them, play an important role for all decisions made by the Load Balancer.

\[
\Pi = \{ \pi_i = P(q_1 = s_i) \} 1 \leq i \leq N
\]

Here \( \Pi \) is an initial state distribution vector. \( \pi_i \) indicates starting probability of the system from state \( i \).

The designer of the computing system has a clear and complete view about each computing node, whole of the system and the program that should be executed on the system. This view will lead the designer to configure the system into the best configuration. The best configuration is the optimal configuration that the system is initiated by. The \( s_1 \) is the first stable state that is configured by the designer and known as the best configuration.

Eq. (6) describes probability of selecting known states at initial time. At first start, we have just one state—first configured state and that is why in our case this set contains one element with probability 100\%. The matrix \( A \) is the matrix of probabilities of transition from one stable state to another after dynamic and interactive events have occurred. The value of \( a_{ij} \)—is the probability of being at state \( j \) at time \( t + 1 \) while being at state \( i \) at time \( t \).

\[
A = \left\{ \left\{ a_{ij} = P(q_t+1 = s_j | q_t = s_i) \right\} \right\}
\]

\[
1 \leq i, j \leq N, \forall t \in [1, T]
\]

Load balancer uses history of transition from one stable state to another for calculating probability of transition of system to the new stable state.

\[
B = \left\{ \left\{ b_j(a_t) = P(a_t | q_t = s_j) \right\} \right\}
\]

\[
1 \leq j \leq N, 1 \leq t \leq T
\]

The matrix \( B \) contains probabilities of selection of node from state \( j \) at given time moment. It indicates usability of node for given request in current state. Optimizing this matrix, we increase resource activity to reach requirement described in Eq. (1). That is why \( B \) can be called as response structure creator. Knowing these parameters gives an opportunity to apply Discrete Time Hidden Markov Model (DTHMM). System known states set \( S \) is proper to state component of HMM. In this case, node set \( V \) can be viewed as objects distributed over states in HMM. Transition history matrix, response structure creator and initial state distribution and discreteness of these parameters over time are compatible to properties of DT HMM.
\[ \lambda = (\pi, A, B) \] (9)

Eq. (9) describes DTHMM. Applying this model, it is possible to find best state set for given request set. On the other hand, it is possible to optimize states for increasing occurrence probabilities of requests in system. It can be described as adapting model to occurred requests. To perform this operation, load balancer should be informed about systems history of states where system have been, and history of resource requests. For the proposed model, when dynamic and interactive event has occurred, load balancer makes new state with current configuration and appends it to states set. Then appends current request set to the request history and optimize model. After optimization, searched best state set for complete request set. As it is the best state set, the last state is the best state for last request.

The proposed model, dynamic and interactive events are not analyzed. Decisions are marked, based on result of dynamic and interactive events. This model can handle dynamic and interactive events in two cases—process forks new process and process communicates with system. Resource request set for a single time moment contains request of all current parallel processes. If new process forked, this set will increase. On the other hand, response structure creator matrix contains current stable response structure for requests. If resource attribute is changed unexpectedly, values of this matrix will change. In second case, load balancer creates new state. For this, state response structure creator evaluated with current values. Because of new state created, dimensions of state transition and initial state distribution changed. The sum of the probabilities of the transitions from one state to all states (including itself) is one. After creation of the new state, it suggested choosing non-zero value for probabilities of transitions to this state and new values for probabilities of the old transitions with following equations.

\[
a_{ij}^* = \frac{a_{ij} \cdot (N-1)}{N}, \quad 1 \leq i < j < N \] (10)

\[
a_{iN}^* = 1 - \sum_{j}^{N-1} a_{ij}^* = \frac{N - (N-1) \sum_{j}^{N-1} a_{ij}}{N} = \frac{N - (N-1)}{N} - \frac{1}{N}, \quad 1 \leq i < N \] (11)

\[
a_{ij}^* = \frac{1}{N}, \quad 1 \leq j \leq N \] (12)

In Eqs. (10)–(12), \(N\) is the new count of states. Eqs. (10) and (11) are proposed for keeping the equality of the sum of the probabilities of outgoing transitions to the unity. As practically not any transition performed from last state, suggested same probability for all outgoing transitions from last state and these fields evaluated as Eq. (12). Initial state distribution also evaluated like Eqs. (10) and (11).

\[
\pi_i^* = \frac{\pi_i \cdot (N-1)}{N}, \quad 1 \leq i < N \] (13)

\[
\pi_N^* = 1 - \sum_{i}^{N-1} \pi_i^* \] (14)

After this evaluation load balancer, optimize model with Baum Welch algorithm (Yang et al., 2017). Then the best state set for the all request history, which ends with current request, is calculated with Viterbi algorithm (Fettweis & Meyr, 1989). The last state of found state set is the best state for current resource request.

When resource attribute changed, if response structure creator matrix which evaluated for this situation, is equals to one of the known state, it means that system entered to fully stability. In this case, load balancer does not create new state just optimizes current model for all request history and chooses last state.
3.2. Pseudo code
1 Evaluate initial configuration of system.
2 Evaluate first state
3 Store request history
4 If dynamic and interactive event occurred
   4.1 If process forked
      4.1.1 Append requests of new process to last request set
   4.2 If resource attribute changed
      4.2.1 Create new state with current parameters of system
      4.2.2 Evaluate new response structure matrix as current values of system
      4.2.3 If evaluated value equals one of the known states proper values
         4.2.3.1 Skip 4.3 and 4.4 steps.
      4.2.4 Else
         4.2.4.1 Append new state to states set
   4.3 Evaluate DT HMM parameters
   4.4 Optimize model
   4.5 Search best state set for all request history
   4.6 Select last state as best state
   4.7 Reload system.

4. Evaluation
To evaluate the proposed model for managing load balancing in distributed Exascale computing systems, a peer-to-peer system is used (Khaneghah, 2017). In this computational system, the system manager uses the notion of computing region for system management. This makes it possible to determine in this type of computing system four areas that are commensurate with the four main sources defined by the operating system.

In order to evaluate the proposed framework, one kind of global activity has been implemented in (Khaneghah, 2017). The WRF (Done et al., 2004; Skamarock et al., 2001) software has been considered as a software tool that requires HPC. This software is based on a global activity of computational resources in the system. Therefore, one kind of global activity is running on the system.

The number of components in the computing system is assumed to be 140 computing nodes. The computing system, which includes 140 computing elements, allows the computational system considered as widespread system for software for testing. The software mentioned above is normally applied on a smaller number of computational components, so that its application on 140 computing nodes makes it possible to determine the state of the software when it is on a widespread system.

Due to the nature of the distributed Exascale computing systems and the need of defining the initial computing system, 60 computing elements were considered for the primary computational system. These 60 computing nodes are in accordance with the basic requirements of the computing processes associated with the scientific and applied program. During the implementation of scientific and applied programs, if the computational processes require new resources to continue the implementation process, the source discovery manager will expand the system and add new resources.

A special version of system manager software (Khaneghah, 2017) has been used on machine number 24 to create dynamic and interactive events, in which the load balancer is based on the proposed model DTHMM ExaLB and the system redistribution and configuration system AMRC.
The hardware design of the appointed computing machine is the same as other computing machines in the system. If, for every global activity, consider a page of activity in accordance with what is stated in (Mirtaheri et al., 2013) considered, in the majority of the times for the implementation of the scientific and applied program, the computational element number 24, the corresponding page of activity there is a national presence. Each other computing element can also be selected as the computing node under consideration.

In computing node Number 24, the System manager has been changed. This causes the system management element in this computing element to manage the three situations of creating processes, communication, and interaction with the system environment that results in a dynamic and interactive nature. For this purpose, in the machine #24, the management element is able to manage processes that are not present in the overall organization structure. B. It manages the communication between the two processes of the global activity organizer, which in the initial structure not related to global activity. (C) Machine No. 12 is linked to another computational machine on which there are processes of the three types of software and software applications mentioned, the car control element number 24, able to manage the relationship between the related process with global activity as well as the corresponding process in the machine, it is outside the computing system that was not considered in the accountability structure of global activity. The system, and consequently the computational element number 24, has been investigated in 50 units of time.

In each element of the computing system (Khaneghah, 2017), two models of AMRC load distribution and DTHMM ExaLB have been implemented. The distribution and configuration manager of the AMRC, at the time of the occurring a dynamic and interactive nature and the change in the workload of the members of the computing system, tries, based on the use of the vector pattern, to compare the workload of the elements of the computing system to the stable state Initial restore. In the AMRC mechanism, from the moment of the dynamic and interactive occurrence and the system enters the non-stable state, the AMRC load balancer, based on the creation of the state of the system and the process, describes how to reconfigure the system and return to the initial state of equilibrium. Do not if the distribution management element of the AMRC mechanism is used, it needs to analyze dynamic and interactive occurrences based on system status descriptor vectors. This mechanism, by describing the state of the system at the time of occurrence of dynamic and interactive events, based on the description of the state about the situation and the comparison of the system description vectors with the system's stable status, while analyzing the dynamic and interrelated phenomena, attempted to reconfigure the system and consequently redistribute the load of the system.

In the DTHMM ExaLB mechanism, from the moment of occurrence of dynamic and interactive event and system state change to non-stable status, the load balancer by considering four indicators of system introduction examines the status of load distribution of the whole system in the non-stable state. The system status is introduced by the four-paired \((S, V, O, Q)\) indicators. In this article, by calculating the values of the \((S, V, O, \text{and } Q)\) let the load balancer has a precise view and be able to make decision while the load distribution changes.

In this mechanism, the load balancer, unlike the AMRC mechanism, does not attempt to analyze the cause of creating a dynamic and interactive nature. In this mechanism, following the occurrence of the dynamic and interactive nature, the system is based on the four concepts mentioned, the distribution manager attempts to describe the status of the system again, if the status of the system is based on one of the previous situations, which is the element of management The distribution is described, in which case the distribution management element based on the triple space \((\pi, A, B)\) describes the situation as a) to enter this situation; b) what status of the distribution has been transferred from the situation; and c) for the redistribution of the specified status of the system, the route followed by the global activity, decides on the load balancing management of
the distribution. Otherwise, it defines the new working condition and calculates the three-
dimensional space \((\pi, A, B)\).

In the DTHMM ExaLB mechanism, the triple element \((\pi, A, B)\) includes the load state of the
computing system in a state that has led to a change in the load state to the current state, the
load position of the computing system in a controlled state and after the current load condition
and the course of the global activity, in which the process of its membership has contributed to the
change in the status of load and the achievement of the current load status. In the mechanism of
DTHMM, ExaLB defines a page based on the model proposed in (Khaneghah, 2017; Mirtaheri et al.,
2013) for each global activity, the line corresponding to the global activity on that page. The
computational elements that are involved in some way in the implementation of global activity are
members of the page corresponding to the global activity. For every global activity, a concept
known as the global line of action can be defined. The global line of action includes all computing
nodes that are part of global activity in each of them are executing. Each global line of action is
necessarily involved in the page of the global activity. The concept of \(\pi\) in the triple element
\((\pi, A, B)\) refers to the concept of global activity. The concept of \(\pi\) points to this that the direction of
global activity, which the process of its members has led to a change in the status of work, has
a pattern and includes what computing nodes.

In the DTHMM ExaLB mechanism, considering that the load balancer is running on the comput-
ing system in which the computational system is running a scientific and applied program requir-
ing HPC system, after the finite number of conditions described by the element DTHMM ExaLB load
Distribution Manager does not deal with the new load distribution situation in the system. The
repetitive nature of the scientific and applied program requires HPC systems that, after a finite
state of the system, the DTHMM ExaLB load distribution manager from time to time will be
repeated in the state of the system description of the system. The number of states describing
the state of the system, followed by the state description of the state of the system, can be any
integer. The dynamic and interactive nature and pattern of occurrence of this nature are such that
it is not possible to decide on the value of this integer. The frequency pattern may be the
occurrence of a dynamic and interactive nature or the number of occurrences that appear in
integer large or small integers, and then the integer, the DTHMM ExaLB load distribution manager
enters the repeated state.

In Figure 1, the number of occurrences that led to the change in the workload in the computing
node num.24 is shown. In Figure 1, the number of occurrences leading to the call to the distribu-
tion management element and the call to the two distribution elements DTHMM ExaLB and AMRC
are displayed.

As seen in Figure 1, the number of occurrences leading to a redistribute the load of the system is
higher than the number of requests that are answered by the AMRC or DTHMM ExaLB load
balancers. The reason for this is due to the nature of the requests that triggered the DTHMM
ExaLB and AMRC load distribution manager. These two load balancers are only active when there is a process by requesting access to a resource within the computing system where the local computing element lacks the ability to respond to it, and this process has a dynamic and interactive nature. While LB Request # includes all the processes that request access to the source that is responsive to the computing system, but the local computing element does not have the ability to respond to it. Figure 1 actually follows the implementation of the scientific and applied program in two successive sequences. In the first run, the distribution management element used to control and manage the dynamic and interactive occurrences of the AMRC mechanism, and in the second implementation, the management element uses the DTHMM ExaLB mechanism.

As shown in Figure 1, the DTHMM ExaLB mechanism, unlike the AMRC mechanism, is able to respond to all dynamic and interactive requests that lead to a change in the workload in the computing system. This is due to the nature of the DTHMM ExaLB mechanism, as compared to the AMRC mechanism. In the DTHMM ExaLB mechanism, the cause and cause of the occurrence of a dynamic nature are not investigated. On the other hand, in the DTHMM ExaLB mechanism for redistribution, the load balancer does not take into account the previous stable condition and return to the previous stable condition. Therefore, the concept of the probability of failure to return to the previous stable condition is not considered.

This issue, while in the AMRC mechanism, is the concept of returning to the previous stable situation, and the load distribution manager tries to use the vectors describing the status of the system and by changing the elements affecting the concept of distribution, the return status of the load and do something to the previous state of stability. This causes, in addition to conventional failures in the implementation of redistributive activities, failures due to the lack of change of the beneficiary elements and the effect on the load of the computing nodes of the system member (or computing area) and returning to the distribution situation work is defined in the state of the previous stable state.

This issue is largely due to the inability of AMRC's load balancer in the analysis of the situation and, consequently, the analysis of the reason for the occurrence of a dynamic and interactive nature, its control and management.

This issue, while the load balancer with the DTHMM ExaLB mechanism, does not attempt to analyze the dynamic and interactive situation and consider the state of the workload of the computing system (or computing area) in the form of the description of the status of the system and in the absence of The description of the state of the system adds a new status to the list of system states and decides on the pattern needed for redistribution based on the triple \((\pi, A, B)\). This makes it possible for the long-range computational system to be examined, in this case, the DTHMM ExaLB mechanism, 100% of the dynamic and interactive requests, has led to a change in the workload and the AMRC mechanism, 74%, of applications dynamic and interactive leads to a change in the status of the workload.

In the case of a computational system review for a long period of time, which includes the acceptable number of stable status, per unit time, the AMRC implemented mechanism is able to respond to 5.68 dynamic and interactive requests that lead to a change less load than the total number of dynamic and interactive requests will lead to a change in the working environment responded by the DTHMM ExaLB mechanism.

Except for time units such as 45, 44, 48, 50, and 51, two load distributions based on the DTHMM ExaLB and AMRC mechanisms use a single model to respond to dynamic and interactive demands that lead to a shift in workload. From this approach, although there is a function difference between the function of the AMRC mechanisms and the DTHMM ExaLB, but the pattern governing the functional function of the two mechanisms is approximately the same. The reason for this is the concept of equilibrium. The AMRC mechanism, based on a backward pattern and the DTHMM
ExaLB mechanism, attempts to redistribute the system by changing the effective indicators on the concept of system status and describing the status of the system.

In the intervals such as 45, 44, 48, 50, and 51, the function of the AMRC mechanism uses a different model, considering the function of the AMRC mechanism in the intervals mentioned above, is the reason for this. The inability of the AMRC to return to the current equilibrium state is based on indicators of the state of the system description and corresponding vectors.

In Figure 2, the number of dynamic and interactive requests that leads to the redistribution, along with the number of new states created by the load balancer, is based on the DTHMM ExaLB mechanism.

As shown in Figure 2, the number of dynamic and interactive requests that require redistribution as well as new status descriptions of the system is presented by the DTHMM ExaLB mechanism. On average, the new state of the new element in computing element 24 is generated at 3.8 times per unit. This is due to the nature of the requests that are answered by the DTHMM ExaLB mechanism. Requests replicated by the DTHMM ExaLB mechanism are both dynamic and interactive events. Dynamic and interactive nature makes it possible for each unit of time to run a scientific and applied program and the occurrence of a request leads to redistribution by the DTHMM ExaLB mechanism, a new situation in describing the system based on quadric spaces (S, V, O, and Q) that existed in previous states. Thus, by repeating the experiment shown in Figure 2, it can be concluded that the occurrence of occurrences and requests of a dynamic and interactive nature, regardless of what causes the occurrence of a dynamic and interactive nature conceptually, it is expected that new system status descriptions will be created. On average, at some point in time from the implementation of scientific and applied programs, about 43% of the workloads and system descriptions based on quadratic spaces are the new status for the system manager based on the DTHMM ExaLB mechanism. In some times units such as 2, 7, and 9, the number of new state events increases from the view of the DTHMM ExaLB distribution manager.

From the point of view of the load distribution manager and based on the DTHMM ExaLB mechanism, changes in the trend of the occurrence of a new situation in short periods of time are not important. From the point of view of the system manager, the DTHMM ExaLB mechanism should reduce the occurrence of a new situation for the system based on the system description over the time of the scientific and applied program, and after a finite interval, the number of occurrences of the situation new in the system to zero. Therefore, from the perspective of the DTHMM ExaLB load balancer, after a certain period of time, no new situation in the system should occur, and all the status descriptions of the system based on quadric space (S, V, O, Q) should be described.

As seen in Figure 2, the number of creation of a new status in the descriptive system of the system is 24 h later. From the first unit of computing element number 24 to time unit 24, the
number of occurrences of the new state of the system description, on average, is about 13 occurrences in the time units of the computational element. However, there are timescales such as 7, 9, 14, 15, and 18 in the first time up to the time unit 24, where the number of occurrences of the new state of the system description is higher than the average.

If the test shown in Figure 2 is performed for the computing node number 24 for the high number of repetitions and the number of units of time examined by the computing node number 24, then it is observed that the average is within a specified time interval and the number of occurrences of the new state of the system description is based on a specific numeric logic.

After this interval, the number of new state events decreases in the description of the system. The reason for this is due to the nature of the scientific and applied programs that run on computational systems, including distributed macroeconomic systems. In this type of scientific and applied programs, including the scientific and applied program used to conduct the experiment in this article, the program has the nature of repeatability. Even parts of the program that are discovering the rules governing the natural event and the processes associated with them are of a dynamic and interactive nature, also follow this.

The occurrence of repetition causes the number of occurrences to occur over a given time period, creating a new state in describing the system on a numerical axis. The size of this number, to a high degree, depends on the number and frequency of the occurrence of a dynamic and interactive nature. After this interval in most of the experiments carried out, it was determined that the number of occurrences of creating the new state of the described system, gradually and gradually began to decrease. In all staggered stairs, there are jumps in the staggered pattern of reducing the occurrence of creating a new state of the system description. This is due to the dynamic and interactive nature of the scientific and applied programs running on distributed macro-scale systems. If the program does not have a dynamic and interactive nature, then there will not be any jumping patterns in staggers of the program, or if there are small jumps available, if the program is missing. While this is a matter of dynamic and interactive nature in scientific and applied programs due to the possibility of dynamic and interactive content occurring at any moment in the implementation of the program, the jumps in stagger reductions also have a number both higher and larger than the traditional ones.

As seen in Figure 2, there are several jumps in a large domain in the first time unit up to the time unit 24. By examining the implementation of the scientific and applied program as well as the changes made to the computing system and the computational elements related to computing element No. 24, it is evident that existing jumps due to the creation of new accountability structures by the source discovery management element, the numerical element of the number 24 is The activation of the resource discovery element in Computational Element 24 makes the structure of the computational system to be modified to create a response structure.

Changing the structure of the computing system requires the need to redefine the system state by quadratic spaces (S, V, O, Q), and from the view point of the DTHMM ExaLB load balancer, the system’s logistic status has entered a new state. From a time, unit of 25, a staggering reduction in the occurrence of a new state of the system’s description occurs, and the mean average number of occurrences of the new state of the system description is 1.8. During a time, unit of 25 to 50 times, in time units such as 33 and 47, there are flipping patterns with a high range.

By analyzing the program running on element 24 and the computational system state, this can be The result is a change in the state of interactions and communication of processes of a dynamic and interactive nature, as well as a change in the pattern of the causes of the creation of a dynamic and interactive nature in the processes of Computational Element No. 24. This is while the DTHMM ExaLB load distribution manager, regardless of the occurrence of a dynamic and interactive nature, and the fact that the occurrence of a dynamic and interactive nature or
a change in its pattern of occurrence, has created patterns of spin in the process of reduction. The step of the number of occurrences of the new description of the status of the system, focuses solely on describing the state of the system or its lack of it, and managing the system load on the basis of triangles ($\pi$, $A$, $B$).

In Figure 3, the time required to implement the redistribution based on the AMRC mechanism and the DTHMM ExaLB, is shown in computing node No. 24.

As seen in Figure 3, the time required to execute the DTHMM ExaLB mechanism if the computing node number 24 for a longer time interval is examined, the average of each algorithm tested at each time interval is 4 Time units are required for redistribution and system reassembly. As shown in Figure 3, the execution time of the DTHMM ExaLB algorithm in the first time unit up to 24 times the average of the time required to run the AMRC algorithm is greater. The AMRC algorithm, for each distribution situation, examines the state of computing element 24 with the stable state and initial configuration. As seen in Figure 3, in the first time unit to the third time unit, the time required for the AMRC algorithm is higher than the DTHMM ExaLB algorithm. The reason for this is the formation of vectors describing the state of labor load, as well as describing the state of computing element No. 24.

The AMRC mechanism considers the occurrence of any occurrence in Computational Element No. 24 as a dynamic and interactive phenomenon. Using the algebra, the condition of the computing element # 24 is the initial configuration of this computing element compares. In case of changing the configuration vector, using the abbreviation and redistribution operators, the load state in the computing element number 24 returns to the initial operating condition. Therefore, during the unit time of 3 to 24, the average time required for the AMRC mechanism is less than the time required to execute the ExaLB DTHMM mechanism. By studying the computational element 24 as well as the application of the scientific and applied program, one can conclude that the existence of jump points in the time unit 3 to the time unit 24 in the implementation of the AMRC mechanism results from the need to repeat the algorithm several times, for describing the vector. The configuration of the computing element number 24 is to accommodate the vector describing the initial configuration of the computing element number 24.

During the time interval from 24 to 50, the time required to run the DTHMM ExaLB algorithm decreases. Given the process of implementing the DTHMM ExaLB mechanism, if the number of times the implementation of the scientific and applied program and the implementation of the test shown in Figure 3 are increased, the implementation time of the DTHMM ExaLB mechanism will reach the zero number. However, according to the DTHMM ExaLB mechanism, it can be argued that the execution time of a mechanism is never equal to zero due to the need to calculate a three-paired ($\pi$, $A$, $B$), but given the nature of the repeatability of the scientific program and functional, from the unit of execution time to the next, the description of the new system is not created and the mechanism of the DTHMM ExaLB is able to decide on the status of how to manage the workload. In the time unit of 24 to the time unit 50, there is a jump pattern.
As shown in Figure 3, although the staggering reduction of the DTHMM ExaLB mechanism occurs faster, and with a staggering reduction, the implementation of the DTHMM ExaLB mechanism is reduced, but it can be Jump jumps, such as time unit 48, at which time the ExaLB DTHMM mechanism increases as compared to the AMRC mechanism. By examining the scientific and applied program as well as the computational element number 24, it is clear that the occurrence of a dynamic and interactive nature has caused the DTHMM ExaLB mechanism to perform its activities to be more time-consuming than the AMRC mechanism. This suggests that, although the DTHMM ExaLB mechanism is suitable for computing systems whose component-delivering elements are in a stable state (Khaneqhoah, 2017) and reduces the time of implementation of redistributive activities. The occurrence of a dynamic and interactive nature makes the DTHMM ExaLB mechanism out of equilibrium. The functioning of the DTHMM ExaLB mechanism in the occurrence of dynamic and interactive occurrences in the system’s equilibrium state is in contrast to the functioning of the AMRC mechanism.

In the AMRC mechanism, each occurrence is a dynamic and interactive phenomenon. In this mechanism, for each occurrence in the change of the load factor of computing element No. 24, the status of the descriptor vectors of the load distribution is compared to the configuration vector and the distribution of the initial stable condition. Take up This issue, while in the DTHMM ExaLB mechanism, as long as there is no description of the new situation in the system, the DTHMM ExaLB mechanism based on the triple (π, A, B) associated with the description of the previous situation, manages the load distribution.

If the computing element number 24 is considered for a long time, then the correlation between the runtime variable of the DTHMM ExaLB mechanism and the number of occurrences that can be managed by the DTHMM ExaLB mechanism is a numerical value of about 0.031. In Figure 4, the linear correlation coefficient between the time of execution of the DTHMM ExaLB mechanism and the number of occurrences managed by the DTHMM ExaLB mechanism is shown.

As seen in Figure 4, in a stable state and a computational node number 24 for a long time interval, between two variables, the number of occurrences that can be managed by the DTHMM ExaLB mechanism and the execution time of the mechanism are indicated. As seen in Figure 4, these two variables are balanced in the computational element number 24 and when this computational node is considered for a long time interval, it has a zero-correlation coefficient. Here R Square is the correlation coefficient, and R² is the square of this coefficient and shows the percentage of variance from the overall variation shown by the regression line.

The cause of this issue is the function of the mechanism. In this mechanism, the main assumption is that it is possible to describe the occurrence based on the description of the previous system’s system and based on the trivial (π, A, B) description of the status of the management system. In the ExaLB DTHMM mechanism, if there is a description of the previous situation, the distribution manager, based on A and B, decides on how the next distribution status is. In addition, if there is a description of the previous situation, the distribution management element on the status of the global activity is based on the π collection of decision-making. Therefore, the DTHMM ExaLB load balancer, if present, describes the status quo, based on the status of the current state of the load (Condition A) and the state of the global activity

Figure 4. Coefficient of correlation between the implementation time of the DTHMM ExaLB mechanism and the number of occurrences managed by the DTHMM ExaLB mechanism.
(variable \( \pi \)) and its effects on what status it must be transmitted (variable \( B \)) so that it can manage the dynamic and interactive behavior.

However, in mechanisms based on the analysis of the nature of dynamic and interactive occurrences such as AMRC, there is a high correlation between the variable of time needed to execute the mechanism and the number of manageable occurrences by the mechanism.

In Figure 5, the correlation between the variable number of occurrences managed by the AMRC mechanism and the time required to implement the AMRC mechanism in stable state is shown.

As shown in Figure 5, there is a correlation of 0.6 between two variables for the time required to execute the AMRC mechanism as well as the number of events that can be managed by the AMRC mechanism at the time of system balancing and checking for a long time. This correlation is based on the fact that the AMRC mechanism, for each occurrence, attempts to calculate vectors describing the load state at the time of occurrence, and describes these situation state vectors with vectors describing the load condition at equilibrium and configuration time compares.

5. Discussion

One of the most important repercussions of occurring dynamic and interactive nature in Distributed Exascale computing environments are computational nodes. Occurring of dynamic and interactive nature causes the load of computational node to be in such a status that mechanism (mechanisms) of manager of load distribution cannot control it. In general, manager of load distribution uses two methods for managing and controlling of workload distribution status as a result of dynamic and interactive environment occurring: dynamic and interactive nature analysis, mechanism definition in line with dynamic and interactive nature, changing the run of scientific and functional plan and managing dynamic and interactive nature of powerful computational element or policy of occurring of dynamic and interactive nature without analyzing dynamic and interactive reason and managing occurrences related to the status change of workload after occurring dynamic nature.

However, using the dynamic and interactive nature analysis policy and defining a mechanism in line with dynamic and interactive manner makes the distribution manager do the allocating processes to resources more carefully, but the main challenge of the mechanisms which use this policy is the high implementation times for analyzing the dynamic and interactive nature and, consequently, defining or redefining the mechanisms related to managing and controlling dynamic and interactive occurrences. The nature of the dynamic and interactive occurrence is such that changes the status of workload, from the viewpoint of load distribution management into a situation in which policy or pattern of facing has not been defined in the manager of load distribution. Therefore, the manager of load distribution should be able to manage and redistribute load by dynamic and interactive nature analysis and by using existing patterns (and by defining new patterns in special cases). Policies depending on the dynamic and interactive nature occurring, unlike those based on the analysis of a dynamic and interactive nature analysis do not analyze and determine what caused the existence of dynamic and interactive nature of the computational element and the change of load status, as a result.
When a dynamic and interactive occurrence causes a change in the status of the workload, even if the distribution manager does not analyze the factors that create the dynamic and interacting nature, the status of the workload of the computing element, as well as the status of the elements of the workload descriptor, are in such a condition that the elements of load distribution management are devoid of mechanisms necessary to deal with. The most important difference between analytical and implementation policies of dynamic and interactive occurrences relates to their pattern to deal with newly made condition for the workload system. In the policies based on analysis, the distribution manager has a precise view of the factors that make the workload condition change from the manageable status of the load distribution manager to a situation that is unmanageable for the load distribution manager.

The existence of a precise view about the causes of unmanageable workload status, provides the possibility for the distribution manager to manage and control new workload status in the system by either changing the analyzed factors and comparing them with indicators describing the mechanism used to manage the load or by changing the trend of global activity and using the ability of the target computing element to manage and control the change in the workload or to change the indicators of the mechanism used for load distribution, affected by the reasons of occurring unmanageable workload status. Conventionally, occurrences in these mechanisms should be analyzed separately for analyzing the workload status. This action is used for dynamic and interactive recognition. The workload manager should be able to have a precise analysis and understanding of the reasons contributing to the existence of dynamic and interactive nature so that it can manage this nature based on its public mechanism.

In mechanisms, based on dynamic and interactive nature analysis, the workload manager tries to control and manage the new status by changing mechanism indicators, dynamic occurrences indicators, or trend change. Dynamic and interactive nature analysis causes the manager of workload can describe the dynamic and interactive nature based on a set of indicators used for describing workload status. This description creates a situation in which the workload manager can analyze the status of the new workload in the system and considers status change of the system as the status change of manageable workload. On the other hand, we have mechanisms and policies of non-analysis of workload status, the load distribution manager is not able to analyze and describe the new status based on indicators describing the workload status because it does not analyze the factors producing dynamic and interactive nature.

Not describing the situation leads the manager of workload to be incapable of changing indicators in the realm of occurrences leading to the status change of workload and in the realm of manager of workload distribution. On the other hand, the absence of an analysis of the nature of the dynamic occurrence and, consequently, of the created status of the workload, makes it possible for the workload manager to use the concept of comparative advantage of computational elements for controlling and managing dynamic and interactive occurrences in these kinds of systems and mechanisms. Workload manager in mechanisms and policies of not describing the nature of occurrence should wait until the dynamic and interactive occurrence looms in the system and then redistribute load based on previous static status or previous management patterns.

In these systems and policies of which DTHMM ExaLB follows, load manager should be able to support the concept of previous static status or similar functional and behavioral patterns. In each of the two mechanisms of the previous stable status or the concept of the functional and behavioral model, the distribution manager describes a concept called the state of the computational system (or even the computational element) from the viewpoint of load distribution element based on a set of indicators. That the selected indicators can describe the computing system from the point of view of the distribution manager refers to the nature and pattern of determining the indicators. In any case, the indicators should be able to describe conditions and statuses of the system, which are based on load distribution, in line with the needed function of the system and from the viewpoint of the load manager. In addition to the system status description indicators,
the occurrence indicators that lead to system state change must also be defined by the distribution manager. These indicators depict events that, if they occur, the system status will change in terms of distribution. These occurrences are based on the centrality of the concept of load distribution, and the central element of controlling these occurrences is the distribution manager.

The element of distribution management will realizes the occurrence of the dynamic and interactive nature in the system in the case of using dynamic and interactive nature occurrence policies, or by examining the status of the system and will try to transfer the distribution status to previous or current stable status state in order to redistribute and manage occurrences or to redistribute the functions and behaviors through studying functions and behaviors of responding to similar status.

The DTHMM ExaLB mechanism uses the policy of dynamic and interactive nature occurrence. In this mechanism, in order to describe the state of the system, the equivalence between the two concepts of the status of the workload system and the concept of system status in the Markov cycles is used. From the perspective of the distribution manager, each system state is considered to be the same as a state in the hidden Markov Model. Although there are several methods for modeling the status of the system from the perspective of the distribution manager, in the DTHMM ExaLB mechanism, the focus is on the fact that the reason for the occurrence of a dynamic and interactive nature for the distribution manager is not a priority for manager of load distribution. The axis of the DTHMM ExaLB mechanism is based on the fact that, firstly, the description of the system state should include information that the load distribution manager in this mechanism needs them in order to manage and control the load. Second, the distribution manager requires knowing that the element of traditional distribution management, based on which model, transfers the system’s load status to it. Load distribution manager needs to know which pattern traditional load distribution manager used in order to transfer the workload of the system to it. It should also become clear provided that status I occurs, to which status the workload of the system will be transferred. This element also should become aware of the fact that on which global activity the status of the load distribution has been defined for the traditional load distribution manager and which road has been used for distributing the load in specified I status- global activity in which dynamic and interactive status has occurred.

The DTHMM ExaLB distribution manager requires the definition of a two-dimensional space (System, State) for the management and control of dynamic and interactive occurrences. The system space indicates how to define system load statuses by the DTMM ExaLB and State manager illustrates the way to extract and manage the change in the workload due to the dynamic and interactive occurrences. Modeling the DTHMM ExaLB distribution manager in this article is based on hidden Markov cycles. In these cycles, the transition from one state to another is only based on how the status of the system before this current one was and what the next status will be and which road has led to current status. In this model, no information is gathered about what has happened in the current situation, which has resulted in the need to change the situation from the current situation to the previous one. In addition, in the mathematical model of the hidden Markov cycles, the patterns of repetition and the concept of homogeneity are used conventionally.

Based on the concept of homogeneity, if a status is repeated in hidden Markov cycles, the triple \((\pi, A, B)\) is not analyzed and, the three-paired \((\pi, A, B)\) of the previous homogeneous cycle is used based on the previous occurrence cycle. This subject has been used in this article for reduction of the time needed to redistribute the load. The nature of scientific and applied programs requires distributed Exascale computing systems in such a way that the program uses a replication pattern after the finite-scale implementation.

Using repetition pattern causes the load management distribution element of DTHMM ExaLB to use described status of the workload and or its repeated version—in which concept of homogeneity and the use of triplet \((\pi, A, B)\) of previous homogeneous states can be applied- to reduce
the time needed for answering. The experiments also indicate that implementation time of the redistributive activities will reduce if the system arrives at the equilibrium situation and the homogeneous situation of the description of the workload is repeated.

In spite of the fact that the existence of homogeneity and the use of triplet $(\pi, A, B)$ of previous homogeneous states causes the DTHMM ExaLB distribution manager to need less time to perform in redistributive activities, this makes the DTHMM ExaLB distribution manager, in the first place, to have completely different behaviors in the repeatability states of the description of the load or non-repeatability ones. Secondly, it makes the correlation between the two concepts of runtime and the number of occurrences managed by the DTHMM ExaLB mechanism to be weak. The most important consequence of these two issues is the high impact of the implementation time of the redistributive activities by the DTHMM ExaLB mechanism in dealing with dynamic and interactive occurrences. In mechanisms that do not use the concept of homogeneity, the frequency of the change in implementation time has a moderate or significant correlation with dynamic and interactive occurrences and events and their effects on the distribution status.

In case of occurrence of dynamic and interactive nature occurrences in the midst of a set of stable and balanced states, this subject causes the time of load redistribution activities not to have humping patterns. However, focusing on the concept of homogeneity in the mechanism of DTHMM ExaLB causes the high sensitivity of time change frequency over the implementation of dynamic and interactive occurrences. As a result, when a dynamic and interactive status occurs between a set of homogeneous states, changes in the time of implementation of redistributive activities will be high. Therefore, the DTHMM ExaLB distribution manager has an appropriate function for scientific and applied programs, in which the time of program and Exascale redistributive computational element’s equilibrium of the distributed macro-scale computing system is higher than the time of program and system imbalance. It also causes the frequency of occurring dynamic and interactive occurrences in time of system’s equilibrium to be higher than homogeneous occurrences. On the other hand, the DTHMM ExaLB load balancer has been considered as an element of dynamic and interactive distribution management of workload in this article. This issue stems from the need for the DTHMM ExaLB distribution manager to extract coherent patterns. In this article, the distribution manager (Khoneghah, 2017), used in the Distributed Exascale computing system is used as the standard distribution manager and extractor of information about the homogenous patterns. The DTHMM ExaLB distribution manager does not attempt to analyze the dynamic and interactive nature. Although this will cause challenges such as sensitivity to the dynamic and interactive nature of the occurrence of redistribution occurrences, it causes DTHMM ExaLB distribution manager to be able to respond to all dynamic and interactive requests. DTHMM ExaLB workload manager reviews the status of workload status descriptions for occurring of each dynamic and interactive occurrence. If the status of workload are not compatible with current status, concept of homogeneity and triplet of $(\pi, A, B)$ are used. Otherwise, it uses workload status as conventional one and conventional workload mechanism for workload management.

The DTHMM ExaLB load balancer distribution manager is considered as a distribution manager for homogeneous situations. This distribution manager can be used to manage dynamic and interactive occurrences influencing on workload status using homogeneous status of load status description in a lesser time than conventional mechanism used for load redistribution. This causes the DTHMM ExaLB distribution manager to be considered as an accelerating distribution manager, along with conventional mechanisms such as mechanisms [1] in distributed Exascale systems.

The proposed model is applicable for centralized Distributed Exascale system. This model requires collecting certain information from nodes to central node. For geographically dispersed highly scalable systems the amount of data transferred in such process is critically huge. Another shortcoming of the proposed model is that it can predict only one new state. And this prediction made based on history of observations. So at the early stages, because of the observation vector $V$ and the passed state vector $Q$ are small, the accuracy of the prediction of the values of the new condition is low.
6. Summary
In distributed Exascale systems, the distribution manager, in addition to managing and implementing redistributive activities, should be able to decide on the pattern used for redistribution. The occurrence of a dynamic and interactive nature in distributed Exascale systems makes it possible to transfer the system workload conditions and computational elements to a state that, they are not able to respond by using load manager mechanism (mechanisms). With this approach, the nature of system status is unknown for load distribution manager from load distribution approach. The DTHMM ExaLB mechanism introduced in this mechanism, as a homogeneous status analysis one, has the capability to does its activity based on the changes that violate the load balance system, instead of analyzing each dynamic and interactive occurrence. The load distribution manager should be able to continue its previous activities, not based on dynamic and interactive behavior, but also based on the new status of the system.

DTHMM ExaLB opens following challenges for future researches: Making this model semi centralized or fully decentralized for decreasing or avoiding data collection from nodes; Making this model being able to predict more than one next state accurately; Increasing accuracy of the predictions in early stages of the system when observation vector and state vector where observation made is small.

Funding
The authors received no direct funding for this research.

Author details
Ulphat Bakhishoff1
E-mail: ulfatbakhishoff@hotmail.com
Ehsan Mousavi Khanegehah2
E-mail: emousavi@shahed.ac.ir
ORCID ID: http://orcid.org/0000-0002-4692-8010
Araz R. Aliev1
E-mail: alievaraz@yahoo.com
Amirhossein Reyhani Showkatabadi1
E-mail: amirhosineyran@gmail.com
1 Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan.
2 Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran.
3 School of Electrical and Computer Engineering, University College of Engineering, University of Tehran, Tehran, Iran.

correction
This article has been republished with minor changes. These changes do not impact the academic content of the article.

Citation information
Cite this article as: DTHMM ExaLB: discrete-time hidden Markov model for load balancing in distributed exascale computing environment, Ulphat Bakhishoff, Ehsan Mousavi Khanegeh, Araz R. Aliev & Amirhossein Reyhani Showkatabadi, Cogent Engineering (2020), 7: 1743404.

References
Alowayyed, S., Groen, D., Coveney, P. V., & Hokestra, A. G. (2017). Multiscale computing in the exascale era. Journal of Computational Science, 22, 15–25. https://doi.org/10.1016/j.jocs.2017.07.004
Chung-Hsien, W., Yan, G.-L., & Lin, C.-L. (1998). Spoken dialogue system using corpus-based hidden Markov model. Fifth international conference on spoken language processing.
Done, J., Davis, C. A., & Weisman, M. (2004). The next generation of NWP: Explicit forecasts of convection using the Weather Research and Forecasting (WRF) model. Atmospheric Science Letters, 5(6), 110–117. https://doi.org/10.1002/asl.72
Elkaham, H., Wagner, C., Grüner, S., Nothdurft, L., & Eppe, U. (2016). “A modular benchmark for evaluating load distribution algorithms.” Industrial electronics society, IECON 2016–42nd annual conference of the IEEE (pp. 4863–4870). IEEE.
Fettweis, G., & Meyr, H. (1989). Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck. IEEE Transactions on Communications, 37(8), 785–790. https://doi.org/10.1109/26.31176
Gaeta, M., Orucioli, F., Raitó, L., & Tomasiello, S. (2017). Fitted Q-iteration and functional networks for ubiquitous recommender systems. Soft Computing, 21(23), 7067–7075. https://doi.org/10.1007/s00500-016-2248-1
Inggi, G., Thomas, D. B., Hung, E., & Luk, W. (2015). Exascale computing for everyone: cloud-based, distributed and heterogeneous. Proceedings of the 3rd international conference on exascale applications and software, (pp. 65–70). University of Edinburgh.
Jairam, N. K., Jagan, A., & Satya Narayana, N. (2015). A novel algorithm for fault tolerant job Scheduling and load balancing in grid computing environment. 2015 International Conference on Green Computing and Internet of Things (ICCGIoT), (pp. 1113–1118). IEEE.
Jiang, Y. (2016). A survey of task allocation and load balancing in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 27(2), 585–599. https://doi.org/10.1109/TPDS.2015.2407900
Khan, S., Nazir, B., Khan, I. A., Shamshirband, S., & Chronopoulos, A. T. (2017). Load balancing in grid computing: Taxonomy, trends and opportunities. Journal of Network and Computer Applications, 88, 99–111. https://doi.org/10.1016/j.jnca.2017.02.013
Khaneghah, E. M., “PMamut: Runtime flexible resource management framework in scalable distributed system based on nature of request, demand and supply and federalism.” U.S. Patent No. 9,613,312. 4 April. 2017.
Khaneghah, E. M., & Mohsen, S. (2016). AMRC: An algebraic model for reconfiguration of high performance cluster computing systems at runtime. The Journal of Supercomputing, 67(1), 1–30. https://doi.org/10.1007/s11227-013-0982-z
Khaneghah, E. M., ShowkatAbad, A. R., & Gharooodi, R. N. (2018). Challenges of process migration to support distributed exascale computing environment. Proceedings
of the 2018 7th international conference on software and computer applications, (pp.20-24). ACM.
Khaneghah, E. M., ShowkatAbad, A. R., Shadnosh, N., Ismailyova, N., Ghahroodi, R. N., Ismailiy, E., Saravani, M. S. N., Saroff, F. T., & Soveili, A. (2018). ExaMig matrix: Process migration based on matrix definition of selecting destination in distributed exascale environments. Azerbaijan Journal of High Performance Computing, 1(1), 20–41. https://doi.org/10.32010/azjhpcc
Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismailyova, N., & Bakhishoff, U. (2018). Challenges of Load Balancing to Support Distributed Exascale Computing Environment. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)(pp. 100-106). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
Kupferman, O. (2018). Automata theory and model checking. In Handbook of model checking (pp. 107–151). Springer, Cham.
Li, S., Shao, X. D., Zhang, Z. H., & Ge, X. B. (2015). Parameter-level data flow modeling oriented to product design process. International Journal of Bifurcation and Chaos, 25(14), 1540037. https://doi.org/10.1142/S0218127415400374
Meddeber, M., & Yagoubi, B. (2016). Distributed load balancing model for grid computing. REVUE AFRICAINE DE LA RECHERCHE EN INFORMATIQUE ET MATHEMATIQUES APPLIQUÉES 12.
Mirataheri, S. L., et al. (2013). A mathematical model for empowerment of beowulf clusters for exascale computing. High Performance Computing and Simulation (HPCS), 2013 international conference, (pp.682-687). IEEE.
Mirataheri, S. L., & Grandinetti, L. (2017). Dynamic load balancing in distributed exascale computing systems. Cluster Computing, 20(4), 3677–3689. https://doi.org/10.1007/s10586-017-0902-8
Mirataheri, S. L., Khaneghah, E. M., Sharifi, M., Minaei-Bidgoli, B., Rasheemi, B., Arab, M. N., & Ardestanii, A. S. (2013). Four-dimensional model for describing the status of peers in peer-to-peer distributed systems. Turkish Journal of Electrical Engineering & Computer Sciences, 21(6), 1646–1664. http://journals.tubitak.gov.tr/elektrik/abstract.htm?id=14276
Najim, K., & Pozynsky, A. S. (2014). Learning automata: Theory and applications. Elsevier.
Pell, S., Abraham, M. J., Kutzner, C., Hess, B., & Lindahl, E. (2014). Tackling exa-scale software challenges in molecular dynamics simulations with GROMACS. International conference on exascale applications and software, Springer, Cham.
Patl, S. S., & Gopal, A. N. (2017). Dynamic load balancing using periodically load collection with past experience policy on Hadoop cluster system. International Journal of Mathematics and Computation, 2(2), 60–75.
Raghu, G., Sharma, N. K., Domanol, S. G., & Reddy, G. R. M. (2018). Memory-based load balancing algorithm in structured peer-to-peer system. Progress in intelligent computing techniques: theory, practice, and applications, Springer, Singapore, 431–439.
Rajkumar, R., Thamarai Selvi, S., & Kannan, G. (2015). Load balancing grid scheduler for the computational grid environment. International Journal of Computer Science Innovations and Technologies, 1(1), 07–19.
Reed, D. A., & Dongarra, J. (2015). Exascale computing and big data. Communications of the ACM, 58(7), 56–68. https://doi.org/10.1145/2699414

Retrieved January 9, 2020 from https://www.deep-projects.eu/applications/project-applications.html
Shang, Y. (2018). Resilient multiscale coordination control against adversarial nodes. Energies, 11(7), 1841. https://doi.org/10.3390/en11071844
Sharifi, M., Hassani, M., Mousavi, E., & Mirtaheri, S. L. (2008). Vce: A new personated virtual cluster engine for cluster computing. 2008 3rd international conference on information and communication technologies: From theory to applications, IEEE.
Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M. (2010). A dynamic framework for integrated management of all types of resources in P2P systems. The Journal of Supercomputing, 52(2), 149–170. https://doi.org/10.1007/s11227-009-0281-x
Sharma, D., & Aggarwal, V. B. (2015). An effective approach for improving performance of load balancing system in cluster computing. International Journal of Computer Applications, 115(7), 21–27. https://doi.org/10.5120/20164-2258
Skamarock, W. C., Klemp, J. B., & Dudhia, J. (2001). Prototypes for the WRF (Weather Research and Forecasting) model. Preprints, Ninth Conf. Mesoscale Processes, J11–J15, Amer. Meteorol. Soc., Fort Lauderdale, FL.
Soltani, N., Khaneghah, E. M., Sharifi, M., & Mirtaheri, S. L. (2012). A dynamic popularity-aware load balancing algorithm for structured p2p systems. IFIP international conference on network and parallel computing, Springer, Berlin, Heidelberg.
Thakur, V., & Kumar, S. (2015). Load balancing approaches: recent computing trends. International Journal of Computer Applications, 131, 43–47.
Thoman, P., et al. (2018). Peter Thoman, Kiril Dichev, Thomas Heller, Roman Iakymchuk, Xavier Aguilar, Khalid Hassanin, Philipp Gschwandtner, Pierre Lemaunier, Stefano Markidis, Herbert Jordan, Thomas Fähringer, Kostas Katinis, Erwin Laure and Dimitrios S. Nikolopoulos. The Journal of Supercomputing, 74(4), 1422–1434. https://doi.org/10.1007/s11227-018-2238-4
Valaich, J. S., George, J. F., & Valaich, J. S. (2017). Modern systems analysis and design. Pearson Education Limited.
Van der Wijngaard, R. F., Georganas, E., Mattson, T. G., & Wissink, A. (2017). A new parallel research kernel to expand research on dynamic load-balancing capabilities. International supercomputing conference, Springer, Cham.
Vangelov, B., & Barahona, M. (2017). Modelling the dynamics of biological systems with the geometric hidden Markov model. bioRxiv, 224063.
Xiao, Y., Xue, Y., Nozarian, S., & Bogdan, P. (2017). A load balancing inspired optimization framework for exascale multicore systems: A complex networks approach. Proceedings of the 36th International Conference on Computer-Aided Design, IEEE Press.
Yang, F., Balakrishnan, S., & Wainwright, M. J. (2017). Statistical and computational guarantees for the Baum–Welch algorithm. The Journal of Machine Learning Research, 18(1), 4528–4580.
Zhang, H., Nemery, P., Qinhe, L. I. N., & Hengelbrock, H. (2019). U.S. Patent No. 10,417,234. Washington, DC: U.S. Patent and Trademark Office.
Zucchini, W., MacDonald, I. L., & Langrock, R. (2016). Hidden Markov models for time series: An introduction using R. Chapman and Hall/CRC.
