Peer-to-Peer Networking and Applications (2021) 14:1021–1043
https://doi.org/10.1007/s12083-020-01067-1

A mathematical model to describe resource discovery failure in distributed exascale computing systems

Elham Adibi [1] · Ehsan Mousavi Khaneghah [1]

Received: 8 September 2019 / Accepted: 29 December 2020 / Published online: 15 February 2021

Abstract
In this paper, a mathematical model is presented to identify the impacts of events with dynamic and interactive nature on the functionality of resource discovery. This mathematical model can recognize those events with the dynamic and interactive nature having an impact on the functionality of resource discovery and thus failure of resource discovery. To extract the mathematical model by which to recognize the failure of resource discovery due to the occurrence of events with the dynamic and interactive nature, a mathematical function that describes the functionality of resource discovery should be determined. To this end, in addition to the description of this function in traditional computing systems, a function describing the functionality of resource discovery is redefined based on the events with dynamic and interactive nature. The functionality of resource discovery during the occurrence of events with the dynamic and interactive nature as well as different ways for the failure of resource discovery due to the impacts of the dynamic and interactive events in distributed exascale computing systems are examined. Determining the type of failure and describing the cause of failure, as well as recognizing an event with the dynamic and interactive nature that leads to failure of resource discovery helps the resource management to prevent failure of resource discovery by changing those features that may cause the failure of resource discovery. The obtained mathematical model is analyzed in two frameworks named PMamut and Cactus. The capability of each framework to recognize events with dynamic and interactive nature based on the usage of the mathematical model are examined. Overall, the model can recognize 52 to 75% of the events that cause a failure of resource discovery.

Keywords Mathematical model · Resource discovery · Events with the dynamic and interactive nature · Failure · Distributed exascale computing systems

1 Introduction

Resource discovery (RD) in distributed computing systems is responsible for searching and finding a resource. Resource discovery is also responsible to have an access to a resource needed by a process that does not exist in the local computing system [1]. Resource discovery tries to find a resource outside the limitations of a system, through which a requester can be able to continue running of the computational process [2]. Generally, RD is used in open computational systems in which the designer of the system has an accurate general overview regarding the events of the scientific application and the necessities of computational processes. However, at the time of the designing of the system, it is not possible to allocate all the required resources to processes. In such computational systems, if a process wants to have an access to resources that cannot be responded to, a load balancer calls RD. Based on a specified mechanism, RD tries to find the resource outside the computational system. Indeed, this mechanism uses a pattern that has not been designed to respond to the requests of the process (i.e. the resource is not in the system) [3, 4].

Open computing systems use meta web as a platform to provide connections between machines [5]. The events of global activity in these systems are uncertain. The behavior and functionality of computing elements of the web and thus the open computing system has an uncertain pattern. The uncertain events of computing systems and RD outside the range of management of the computing system may cause the failure of RD [6].

✉ Ehsan Mousavi Khaneghah
EMousavi@Shahed.ac.ir

Elham Adibi
Elham.adibi@Shahed.ac.ir

[1] Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran

In traditional computing systems such as grid computing systems, the request that leads to calling of RD is not changing from activation of RD in response to the request of the process, meaning that the state of the request is not changing. If a factor in the system has a specified impact, this will have remained unchanged during the execution activities related to RD. In other words, the state of the request and factors influencing the request has remained constant during activities related to RD. As a result, RD mechanisms focus on the way to find a resource and give permission to the process to have access to the resource. The mechanism of RD reveals the pattern of the designer of the system to effectively respond to the requests of the processes that cannot be responded to by the local computing system. In such computing systems, RD tries to find a resource that is compatible with the request of the process. Thus, the main task of RD in these computing systems is based on finding and matching the resource [7].

The definition of RD based on finding and matching the resource may lead to failure of RD by any factor that influences finding and matching the resource. Thus, in distributed exascale computing systems, by examination of factors that influence the execution of activities related to resource discovery, those factors that cause the failure of resource discovery can be defined.

As the states of computing elements that are influencing the requestor are influenced by the request are constant and the fact that constraints governed on the request are not changing, factors have defined that lead to failure of RD outside of distributed computing system [7].

Due to the occurrence of the events with the dynamic and interactive nature in distributed exascale computing systems, it is possible to change the state of the system and thus the state of the requester at any time during the execution of a scientific application. Events with the dynamic and interactive nature may lead to changes in the state of computing elements of the system and thus changes in factors influencing the request of the processes or are influenced by the request. These changes may cause the computing system would be able to respond to the initial request. As the state of distributed exascale computing systems may change, failure of RD and the reasons behind that are more complex than the failure of RD in traditional computing systems [8]. In distributed exascale computing systems, in addition to the responsibility of resource discovery for finding the resource, it should create a responding structure. This means that matching the discovered request and the resource is not the only responsibility of resource discovery.

Based on the necessities of the process, RD may define a global activity to respond to the request of the process. Thus, the pivotal element is changing from finding a resource to the creation of a responding structure. The created responding structure is a dynamic structure, which is caused by events with dynamic and interactive nature. As a result, the failure of activities related to RD is defined as a failure in the responding structure [9, 10].

In traditional computing systems, in most cases, the concept of failure of activities of resource discovery is related to the reversal of the spatial/temporal constraints of the response to the request. In such systems, if the resource cannot be found in the specified time, the functionality of resource discovery management is failed. Nevertheless, in distributed exascale computing systems, due to different functionality of resource discovery management, the occurrence of events with the dynamic and interactive nature might create conditions in the system that make it impossible to continue the execution of activities related to resource discovery management with the spatial/temporal constraints.

In this paper, the functionality of RD is presented by examining a mathematical model introduced here. Besides, it will be discussed in which states the failure of RD in traditional computing systems may occur. Based on those states, failure of RD in distributed exascale computing systems is examined. The impact of the events with the dynamic and interactive nature on the functionality of RD and the definition of the pattern of failure of RD in distributed exascale computing systems are also discussed. Based on the definitions presented in this study, a mathematical pattern for a description of the failure of RD in distributed exascale computing systems is introduced.

2 Related work

In this section, activities related to the failure of activities of resource discovery management are discussed. For each activity, individual characteristics of that activity and the possibility of its application in distributed exascale computing systems are also discussed.

Scalability characteristic in peer-to-peer distributed systems causes computing elements to be added to or removed from the computing system. In [11], the application of blind mechanisms for resource discovery is described as an inefficient method. This is since blind searching requires higher bandwidth and causes network traffic. Resource discovery might not be able to find the resource is the searching space. In [11], a searching mechanism for resource discovery is introduced that estimates the responding state of computing elements to the received requests. In this mechanism, each computing element has a table to store information of adjacent resources and in case of any changes, the stored information is updated.

Due to the occurrence of the events in scientific applications, the rate of failure is high in distributed exascale computing systems. The rate of failure can be described by the dynamic information of resources and membership of nodes in the system. The impacts of failure can be discussed in two

areas: (1) resources; and (2) nodes. If resources fail, those nodes that release their description might be needed to update their local information. Thus, failure of resources is considered as changes in characteristics of resources that are caused by the dynamic events of the information of resources. If nodes fail, not only these resources become unavailable, but also the impact on the requests of other nodes. Thus, the failure of nodes is considered as exits of the node from membership of the system. This is unpredictable and leads to failure of RD [3, 8]. In [12], changes in functionality of RD due to the occurrence of events with the dynamic and interactive events, as well as changes in the state of the system and resources are discussed. In [12], conditions that lead to the failure of RD are discussed, and a mathematical model for RD before and after the failure of RD is introduced.

In [2], challenges of the functionality of resources discovery management are described. In [2], communication between processes in and out of the computing system is introduced as a factor for the creation of changes in the necessities of the process activating resource discovery management. Following any change in the state of the request, the system or the process can change constraints of the request or can create a new requirement for the process. The abovementioned changes can influence the functionality of resource discovery and activities of resource discovery might fail.

By increasing the necessity of communication with other systems, some mechanisms are required for configuration and re-configuration of these communications. Applications that require the HPC are such that they are required to re-organize communications with other systems during the execution of the program. The nature of these applications is in a way that unpredictable events might create during the execution of the program and if there is no mechanism to find new systems, the functionality of the system fails. In [13], a framework based on mathematical rules is introduced to re-configure communication between systems. In [13], a distributed system is considered as an organization and it is examined how this organization can communicate with other organizations without a problem in the functionality of the system. In [13], the concept of events with dynamic and interactive nature is examined. The impacts of such events on different elements of the system and possible changes are also described. The mathematical model presented in [13] is based on a non-Euclidean space.

Today, the development of software systems is based on the build-up of services executed on distributed platforms. The necessities of processes are changing during the execution of the program. Thus, during the design of the system and applications, the dynamic necessities of processes should be considered. Using the process pattern is an efficient method for the development of the dynamic system. In [14], a process pattern is introduced to improve the efficiency of the functionality of the system.

In [7], a mathematical model based on the derivative of a matrix is introduced to analyze the functionality of resource discovery in distributed exascale computing systems. It is stated in [7] that the concept of the process should be considered as part of the global activity in distributed exascale computing systems. It is also stated in [7] that resource discovery can only manage dynamic characteristics of distributed exascale computing systems if it contains mechanisms to analyze changes in the request during the execution of the program. Besides, resource discovery should have mechanisms to analyze the type of the request, such that it can establish the best match between the requester and the responder at each time. For supporting mechanisms specified in this article, by computing the instant derivative of the effective variables on the functionality of resource discovery, it should acquire capability for the management of events with the dynamic and interactive nature.

If RD is conducted by flooding mechanism for propagating searching messages to other machines, traffic may be increased in the network. To prevent that, an index called time to live is used which indicates the maximum number of machines that can be released from the requester. The passage from each of the computational elements leads to a one-unit reduction of this index. If the index becomes zero or the requested resource cannot be found, RD ends, leading to failure of RD [15, 16].

In [2, 7, 12], conditions for failure of activities related to resource discovery in distributed exascale computing systems are defined. In these studies, it is tried to introduce mechanisms to empower resource discovery for supporting the dynamic system and for creating changes in the state of processes. In none of [2, 7, 12], a mathematical model was presented to describe different forms of failure in the functionality of resource discovery. In this article, in comparison to the abovementioned articles, in addition to analysis and examination of the concept of events with the dynamic and interactive nature inside the system based on the vector algebra, the concept of failure, reasons of failure, and the pattern to deal with the factors that influence the functionality of resource discovery are examined.

3 Basic concept

Due to the functionality of RD outside of the system, a pattern should be defined whose activities are outside of the system. As a result, basic concepts that define RD are different from those of other active units for resource management. In what follows, the definition of RD and its functionality, as well as failure of RD are discussed. Besides, the impact of events with the dynamic and interactive events on the functionality of RD and the reason behind the failure of RD are explained. As a

result, a better understanding of the concept of failure of RD in distributed exascale computing systems will be obtained.

3.1 Resource discovery in traditional computing systems

In this section, different definitions of RD in traditional computing systems are discussed. In distributed exascale computing systems, any factor that affects the functionality of RD while considering constraints and limitations of the request, such that the resource cannot be found, is considered as the failure of RD. Thus, the presented definition can deal with the failure of RD if it contains a mechanism for that.

A) In a distributed network of computers, it is usual for a subset of machines to cooperate to perform a common task. The first step in such cooperation is for machines to learn about the existence of each other. Harchol-Balter et al. call this first step the resource discovery [17].

Definition A contains the functionality of RD. Based on this definition, machines of a system should be aware of the resources of other machines of the system. In this definition, the main task of RD should be gathering information. The main advantage of this definition is its simplicity, such that only collecting information is emphasized, which itself is a branch of finding that is related to RD. Thus, the failure of RD means that gathering information has not been successful. In definition A, a mechanism by which to face this kind of failure is not considered.

B) The ability to locate resources that satisfy a set of requirements mentioned in a query (request) [18, 19].

Definition B describes RD given the requesting process. After activation of RD, it tries to search and find the requested resource outside of the computational system. The time and location constraints and limitations and the type of the requested resources are determined in the requesting process. Resource discovery should find a machine that can satisfy all constraints, otherwise, RD fails. In this definition, a mechanism by which to face this kind of failure is not considered.

C) Resource discovery is the process of an entity in a network (i.e. a client) is automatically made aware of accessibility to services or desired devices (resources). More exactly, resource discovery is a mechanism for dynamic referencing to resources in the network. These references are handles or information with which a client can use to contact a resource further [20].

Similar to definition A, definition C contains the functionality of RD. In this definition, RD is based on gathering information about the resources of computational elements. The most important feature of definition C is the dynamic relationship between the resource and the requesting process. Based on this definition, gathering information about states of existing resources is done by resource management. This information is then analyzed to respond to the request. The information is gathered in two ways: (1) resource management gathers information in a specific period; (2) based on the occurrence of an event in the system, the existent information is being updated. Thus, RD is done based on the last state of resources. If the frequency of changes in the state of resources is low, these two methods can be used effectively and relatively fast. Considering the dynamic events of resources and providing conditions of the requesting process may cause the failure of RD. In this definition, no mechanism is provided to prevent the failure of RD.

D) Resource discovery is one of the essential challenges in Grid, which discovers appropriate resources based on the requested task. Certain factors make the resource discovery problem difficult to solve. These factors are the huge number of resources, distributed ownership, heterogeneity of resources, resource failure, reliability, dynamicity, and resource evolution [2, 21].

Definition D states that RD is conducted in a dynamic environment based on constraints of the request. The dynamic events of the environment and variability of computing elements outside of the system can lead to failure of RD. In this definition, one of the factors that complicate RD is the unavailability of resources. In this definition, no mechanism is presented for the management of the failure of RD.

E) Searching and locating candidate resources considered which are suitable for a job. It must be noticed that processing environments' constraints are specified in each task. On the other hand, resource discovery must be performed in a reasonable time regarding the high dynamicity and scalability of the environment [22].

Similar to definition B, this definition represents the responsibility of RD because of the requesting process and the machine containing the resource. In this definition, the dynamic events of machines are emphasized. The frequency of the occurrence of changes in the state of machines containing distributed peer to peer systems is higher than those of traditional systems. This can be one of the reasons for the failure of RD. In this definition, a mechanism dealing with the failure of RD is not considered.

F) Important issue in resource management is the efficient assignment of resources to clients [23].

Definition G implies that resource management conducts the allocation of a resource to the requesting process. In distributed systems, after sending a request by a process, resource management processes the request and tries to allocate an appropriate resource. A load balancer in traditional computing systems and resource management in distributed exascale computing systems do this activity. If any event causes unsuccessfulness of the resource allocation, RD fails. In this definition, no mechanism is introduced for the management of failure of RD.

G) One of the main tasks of RD is providing a consistency between the request and the resource. The "Match Algorithm" which is a flexible mechanism for resource management can do this. This mechanism is useful during the design, deployment, and improvement of the distributed system [24].

This definition describes a matching operator in resource management. Resource management should be able to identify any of the constraints of the request. Resource management is searching for the resource based on characteristics of the request. If resource management cannot match the request and the resource that is found, failure due to inconsistency occurs. In definition H, it is not discussed how to manage this type of failure.

H) To prevent a high number of requests to grid nodes, an index called time to live is used. Resource discovery continues as long as this index is not zero or the requested resource is not found [25].

A definition I represent the functionality of RD in terms of the time constraint of the request. Time constraint can be defined by an index called the time of live. The value of this index indicates a range of the propagating request from the requesting machine. If this index becomes zero, RD stops or fails. In this definition, the management of failure of RD is not considered.

Each of the definitions presented above describes conditions that disturb activities related to RD. Based on the above definitions, it can be concluded that the possibility of failure increases when the dynamic events of the system and changes in the system increase. The events of practical programs in distributed exascale computing systems are in such a way that the state of the request and the process are changing during the execution of a program. Thus, RD needs to apply more complex mechanisms to manage these conditions. As it is known, failure of RD leads to less performance of the computational system and an increase in the response time to requests. Thus, knowing the way that failure of RD occurs can provide a better view or management. Definitions A to I have described the failure of RD in traditional systems, while conditions lead to failure of RD in distributed exascale computing systems are not considered in these definitions.

3.2 The functionality of resource discovery in traditional computing systems

In this section, the functionality function of resource discovery management in traditional computing systems and those factors that influence this function are introduced. In contrast to closed traditional computing systems, in open traditional computing systems not all necessary resources at the time of designing the system and the start of the scientific application are available for the load balancer. In such computing systems, after the start of the scientific application, if a resource is required that is not existent in the system, the load balancer calls RD. The required resources for the continuation of the scientific application are found by RD and will be available for the load balancer. The load balancer allocates the resource to the requester. The designer of the system defines characteristics of the required resources for the continuation of the scientific application. Based on these characteristics and the request of the load balancer, RD finds the resource [9]. The pivotal element for RD is consistency between the characteristics of the resource and the request. The consistency function can be described as follow:

$$\begin{aligned} y &= F(\textit{resource Discovery}) \\ &= \left(\textit{Resource} \rightarrow \textit{Process}_{\textit{requirment}}\right) \\ &= \sum_{i=1}^{k} \beta_i \alpha_i - \sum_{j=k+1}^{n} c_j \alpha_j \approx 0 \end{aligned} \tag{1}$$

As Eq. 1 shows, the consistency that has taken place by RD can be defined as mapping space characteristics of the resource on space characteristics of the process, as well as in vector form. In vector form, if V denotes the space of the resource, W denotes the space of the request and T denotes the functionality of RD, T is a map from V to W. In this case, $\{\alpha_1, \alpha_2, \ldots, a_k\}$ can be regarded as constituent elements of the request. All constituent elements of this space contain the requested resource. Vectors such as $\{\alpha_{k+1}, \ldots, \alpha_n\}$ should be existed in the space of the resource to define the functionality of the resource.

In Eq. 1, the consistency is taken place by RD when $Alpha_{cost} = \sum_{j=k+1}^{n} c_j \alpha_j$ and $Alpha_{Time} = \sum_{j=k+1}^{n} \beta_j \alpha_j$ are equal in the same direction. In traditional computing systems, the functionality of RD is not acceptable when either the cost or the time has taken for RD is high and failure of RD occurs. If based on Eq. 1, two vectors $Alpha_{cost}$ and $Alpha_{Time}$ become equal, consistency by RD has been successful.

4 The functionality of resource discovery in distributed exascale computing systems

In this section, the functionality function of resource discovery management in distributed exascale computing systems is examined.

If at the time of t = Alpha, the request Beta is created by the computing process for which the load balancer cannot respond to that, RD is being called. One of the main differences between the request in traditional and distributed exascale computing systems is the nature of the Beta request. In traditional computing systems, the beta request is one-dimensional. Given RD, one-dimensionality of the beta request implies that the request is only about one specific type of resource. In this case, RD should find a resource, which is 100% consistent with the requesting resource by the process [12].

In traditional computing systems, the state of the beta request is constant during the execution of activities related to RD, meaning that the type, nature, and functionality of the request remains constant. On the other hand, in distributed exascale computing systems, there is no constraint regarding the constant state of the beta request. The occurrence of any event with the dynamic and interactive nature in the process may change the state of the requests in a way that it influences the type, nature, and functionality of the request in global activity. The state of the global activity in which the beta request is part of it when the request cannot be responded is shown in Fig. 1.

In Fig. 1 it is assumed that the responding structure is created based on the necessities of the processes and ability of computing elements. During responding to the necessities of the global activity, the event with the dynamic and interactive nature causes the beta request to be created, which cannot be responded by the local computing system [2, 12]. As shown in Fig. 1, in this situation RD should create a new responding structure for the bet request. Part of the global activity, which is not dependent on the beta request, is managed by the load balancer and continues at the local computing system.

As the beta request is part of the global activity, the space of the beta request is two-dimensional (i.e. global activity, local state). The global activity dimension indicates interaction and communication of the beta request and the global request, as well as their impact and influence on each other [8]. The local state dimension also indicates interaction and communication of the beta request and the process of sending the request, as well as their impact and influence on each other. Given RD, failure of the beta request is the outcome of not responding to the request in any of the two spaces. Given RD, the beta request can be described as follows

$$Request_{beta} = L\left(Request, Request_{Space}, Request_{Vector}\right) \quad (2)$$

As can be seen in Eq. 2, the beta request is in the form of a vector function with two spaces of the Request and $Request_{space}$, as well as $Request_{vector}$. The vector space of request indicates the pattern governs on the request, which is in the form of (time, location, dependency), in which time indicates time constraints and necessities of the request, location indicates location limitations and necessities of the request and dependency indicate limitations of the request relative to the global activity. The conditions under which the response is performed are defined in the dependency space.

The $Request_{space}$ indicates generating space of the request. In distributed exascale computing systems, this space is two dimensional considering two space generator of the global activity and local machine. In distributed exascale computing systems, the two-dimensional space (global activity, local state) is considered as the vector space $Request_{space}$. The global activity and local state spaces are defined at the space generator of the answer. The space generator of the answer contains all elements and spaces that can respond to one of the requests or part of the one request. Given RD in a distributed exascale computing system, each computing element is used

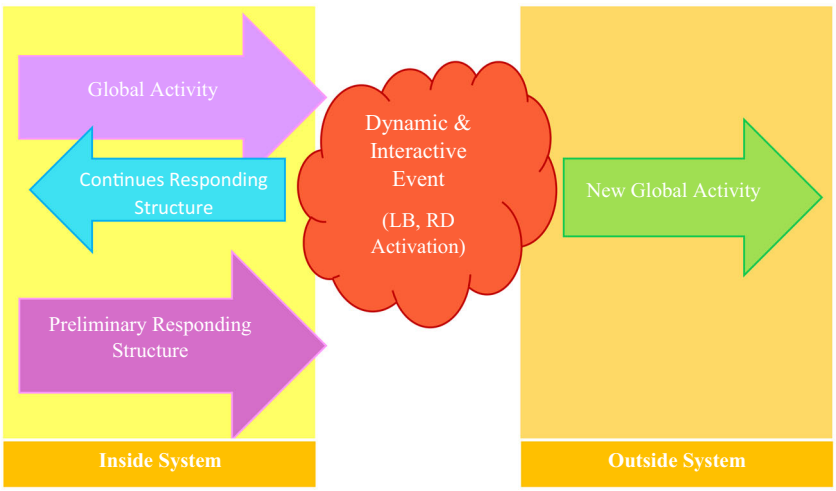

Fig. 1 The impact of an event with the dynamic and interactive nature on a distributed exascale computing system

based on the fact that whether or not it can respond based on the space generator of the answer.

The generating space of $\text{Request}_{\text{vector}}$ indicates a breakdown of the request to sub-requests, which should be responded by the $\text{Request}_{\text{space}}$. Members of $\text{Request}_{\text{vector}}$ are elements that cannot be broken down. In distributed exascale computing systems, resource management of each element has its independent operating system. As such, the space generator of $\text{Request}_{\text{vector}}$ can be regarded equivalent to the classification of resources because of the operating system. In such conditions, the space generator of $\text{Request}_{\text{vector}}$ is regarded with four characteristics (IO, process, file, memory).

According to Eq. 2 and considering events with the dynamic and interactive nature in the computing element of the beta request, a concept called the degree of consistency of the request and the response is defined. This concept causes RD to be able to determine its functionality at a specific time. Eq. 2 describes the definition of the beta request based on vectors of the request and consideration of the degree of consistency. In Eq. 3, the function of RD at a certain time is specified:

$$f(\textit{Resource Discovery}) = \sum_{k=1}^{m} \underset{\textit{Attribute}_{\textit{vector}_k}}{\left((\textit{Resource}_{\textit{Vector}} | \textit{Attribute}_{\textit{vector}_k}) / \| \textit{Attribute}_{\textit{vector}_k} \|^2 \right)} * \quad (3)$$

As can be seen in Eq. 3, the RD function is a vector. Given RD, each resource of the system as well as the request of the process can be described based on a set of vector characteristics. The nature of the request of in distributed exascale computing systems follows a discrete pattern. If the request is in such a way that all its parts cannot be responded by a machine, the RD function in Eq. 3 is presented as sigma.

Based on the classification of resources because of the operating system as a local manager, the same classification can be applied for RD. Thus, RD considers each resource as an orthogonal sequence of four vectors $\text{IO}_{\text{Vector}}$, $\text{File}_{\text{Vector}}$, $\text{Memory}_{\text{Vector}}$, and $\text{Process}_{\text{Vector}}$. Each request vector can be described as an orthogonal sequence of the four vectors.

If in Eq. 3 k = 1, the traditional RD pattern has been used. In this pattern, RD tries to find a resource for the request of the process by which all the necessities of the requester can be responded to or the request of the process follows the pattern of traditional requests. If in Eq. 3, k > 1, RD has been conducted a global activity. Based on what is governed on distributed exascale computing systems, the global activity cycle might be considered as part of the system. In Eq. 3, instead of $\text{Resource}_{\text{Vector}}|\text{Attribute}_{\text{vector}k}$, Eq. 4 can be used.

$$(\textit{Resource}_{\textit{Vector}} | \textit{Attribute}_{\textit{vector}_k}) = \frac{1}{4} \sum_{n=1}^{4} i^n || (\textit{Resource}_{\textit{Vector}} | i^n \textit{Attribute}_{\textit{vector}}) || \quad (4)$$

Equation 4 is rewriting of the $\textit{Resource}_{\textit{Vector}} | \textit{Attribute}_{\textit{vector}_k}$ based on the consideration of characteristics of the scalar product. Vector i depicts a unit vector for each resource and thus for each requesting process. The direction of the vector is in line to respond to the request of the process and its magnitude is 1. At the beginning of the formation of the request, the direction of the vector is not known. At the first computational element that part of the request is responded to, the direction of the unit vector of a resource is similar to the direction of the corresponding resource in the cubic of the responding machine. By replacing Eq. 4 in Eq. 3, Eq. 5 is obtained which describes RD functional vector.

$$f(\textit{Resource Discovery}) = \left[\sum_{k=1}^{m} \left(\frac{1}{4} \sum_{n=1}^{4} i^n || (\textit{Resource}_{\textit{Vector}} | i^n \textit{Attribute}_{\textit{vector}}) || / \| \textit{Attribute}_{\textit{vector}_k} \|^2 \right) * \overrightarrow{\textit{Attribute}_{\textit{vector}_k}} \right]_{\textit{Request}} \quad (5)$$

As can be seen in Eq. 5, the first part of the vector shows the functionality of RD. This vector is described based on the space of the scalar product. The $\textit{Resource}_{\textit{Vector}} \mid i^n \textit{Attribute}_{\textit{vector}}$ the vector represents the capability or incapability of the computational element in responding to the resource request. The power of i in this vector represents the dimension of the vector in the Tesseract cubic [30]. Quadratic form of $\| \textit{Attribute}_{\textit{vector}_k} \|$ indicates the importance of the type of resource for the requesting process. This coefficient leads to modification of the vector magnitude obtained from $\textit{Resource}_{\textit{Vector}} \mid i^n \textit{Attribute}_{\textit{vector}}$ and fits in with the importance of the resource for the process. Multiplication of $\sum_{k=1}^{m} \left(\frac{1}{4} \sum_{n=1}^{4} i^n || (\textit{Resource}_{\textit{Vector}} | i^n \textit{Attribute}_{\textit{vector}}) || / \| \textit{Attribute}_{\textit{vector}_k} \|^2 \right)$ at $\overrightarrow{\textit{Attribute}_{\textit{vector}_k}}$ causes the state of the requesting process to be specified after the response by the computing element. This vector shows changes that have been taken place in the vector ($\overrightarrow{\textit{Attribute}_{\textit{vector}_k}}$) related to the request of the process about the specific computing element k. Multiplication of the two vectors causes the state of the request to be expressed after the response by the computing element k.

Given RD, each resource can be defined as a sum of four-vectors describing states of resources: I/O, Memory, File, and Process [26]. For a description of each resource, RD creates a Tesseract [27]. Each side of the 4-dimensional cubic represents one of the vectors. During the formation of the system, each vector doesn't have to be located on which side of the Tesseract. If the resource regarding any of the four resources of I/O, Memory, File, and Process responds to the request of the process, its value increases in a positive direction. If the

resource cannot respond to the request of the process regarding each of the four resources I/O, Memory, File, and Process, its value increases in the negative direction. When the resource receives a request, this request is separated into four vectors, and the response depends on the directions and magnitudes of these vectors.

If the resource can respond to one or more than one type of the resources I/O, Memory, File, and Process, vector magnitudes corresponding to I/O_{Vector}, $Memory_{Vector}$, $File_{Vector}$, and $Process_{Vector}$ represent the period that the resource can be in the access of the requesting process. If the resource does not respond to the request for one or more than one type of resource I/O, Memory, File, and Process, vector magnitudes corresponding to I/O_{Vector}, $Memory_{Vector}$, $File_{Vector}$, and $Process_{Vector}$ represent the time has been taken by RD to find a resource.

In traditional computing systems, the request is one dimensional [28]. Given the machine containing the resource, when RD wants to find a resource, only in one of the abovementioned four dimensions requests to use a resource. Thus, in traditional computing systems when a resource responds to the request regarding one of the resources I/O, Memory, File, and Process, the state of the corresponding vector in the other three resources is constant. If a resource is requested in traditional computing systems, only a vector related to that resource will change, while magnitudes of other vectors do not change. In contrast, in distributed exascale computing systems, the other three vectors are changing in the negative direction for the period has taken by RD to find a resource.

Based on Eq. 6, the sum of four-vectors I/O_{Vector}, $Memory_{Vector}$, $File_{Vector}$, and $Process_{Vector}$ is represented by $Resource_{Vector}$.

$$\begin{aligned} Resource_{Vector} &= Process_{Vector} \oplus Memory_{Vector} \oplus File_{Vector} \oplus IO_{Vector} \\ &= \pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector} \otimes Memory_{Vector} \otimes File_{Vector} \otimes IO_{Vector}) \end{aligned} \tag{6}$$

As can be seen in Eq. 6, considering a compound form of these vectors, $Resource_{Vector}$ is the cross product of the vectors that constitute the resource. The resulting vector ($Resource_{Vector}$) is the describer of the state of the resource. In distributed exascale computing systems, each request can be about one type of resource or a combination of four types of resources. The machine that owns the resource can respond to the request of the process for each of the four resources or a combination of them based on the concept of changing functionality [26, 29]. If the request is about more than one type of resource, RD should be able to consider challenges related to compound states of different types of resources. Changing the compound state of the request might lead to the inability of the resource to respond to the request. Given RD, for each resource, a 4-dimensional cubic can be described. In each time, the 4-dimensional cubic of each resource has four vectors corresponding to four types of resources. $Resource_{Vector}$ in each time is created by RD. Given RD, this vector indicates the capability of each resource in responding to the requests of the processes activating RD. $Attribute_{vector_k}$ represents the necessities of the request for each of the four defined resources. This vector can be described as:

$$\begin{aligned} Attribute_{vector_k} &= Process_{Vector} \oplus Memory_{Vector} \oplus File_{Vector} \oplus IO_{Vector} \\ &= (Process_{Vector} \otimes Memory_{Vector} \otimes File_{Vector} \otimes IO_{Vector}) \end{aligned} \tag{7}$$

As can be seen in Eq. 7, for each request that RD is called, Tesseract cubic is created. This compound space is the result of compounds of cross products of I/O, Memory, File, and Process.

In Eq. 7, space of the characteristics of the request is as the compounds of cross products of the necessities of the process k. Each vector results from a cross-product that represents the necessities of the request for each type of the defined resource in a distributed exascale computing system. In Eq. 7, if the dynamic and interactive events do not occur in the requesting process, it is not required to consider all state's results from cross products of vectors. In this condition, there is only one state that can be responded to by RD. Due to the occurrence of events with the dynamic and interactive nature in distributed exascale computing systems, all states of cross products are considered. Thus, Eq. 7 represents the state of $Attribute_{vector_k}$ in the distributed exascale computing system.

4.1 The functionality of resource discovery for events with dynamic and interactive nature

In this section, the impacts of events with the dynamic and interactive nature of the functionality of resource discovery management in distributed exascale computing systems are discussed.

Equations 2 to 7 describe states of the request, resource, and functionality of RD at a specific time before the occurrence of events with the dynamic and interactive nature. Given RD, events with dynamic and interactive nature lead to changes in the state of the requesting process. Changing the state of the request affects the characteristics of the request and functionality of RD. The impact of events with the dynamic and interactive nature on the request causes Eq. 2 changes to Eq. 8:

$$Request_{Beta}{}^{Time}_{D\&I} = T\left(Request_{Beta}\right) = \sum_{p=1}^{n} X_p Y_p \tag{8}$$

In Eq. 8, T is a linear map that takes place from $Request_{Beta}$ on $Request_{Beta}{}^{Time}_{D\&I}$. This linear map is the impact of events with the dynamic and interactive nature of the space of the $Request_{Beta}$. In Eq. 8, X_j is the space coordinates of $Request_{Beta}{}^{Time}_{D\&I}$ after the occurrence of events with dynamic and interactive nature. Y_i is the space coordinates of $Request_{Beta}$ before the occurrence of events with dynamic and interactive nature.

To examine the impact of events with the dynamic and interactive nature on the space of the characteristics of the request, T can be regarded as a linear operator on the space of $Attribute_{vector_k}$ with the finite dimension. Characteristics of the request under the T mapping changes from what shown in Eq. 7 to Eq. 9:

$$Attribute_{vector_k}^{Time}{}_{D\&I} = \mathrm{T}(Attribute_{vector_k}) = \left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector}; Attribute_{vector_k})\right] \oplus \left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector}; Attribute_{vector_k})\right] \oplus \left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector}; Attribute_{vector_k})\right] \oplus \left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector}; Attribute_{vector_k})\right] \quad (9)$$

Characteristics of the request are part of the space of the request in Eq. 2 which should be responded to during execution of activities related to RD. Considering the definition of the space of the request based on Eq. 7, T can be regarded as N vector operator. The vector operator for responding in a space with finite dimension is $Attribute_{vector_k}$. As stated in Eq. 7, each resource has four types of characteristics. Thus, with four-vectors $\mathrm{I/O}_{\mathrm{Vector}}$, $\mathrm{Memory}_{\mathrm{Vector}}$, $\mathrm{File}_{\mathrm{Vector}}$, and $\mathrm{Process}_{\mathrm{Vector}}$ in $\mathrm{Attribute}_{\mathrm{vector_k}}$ and N vector operators according to Eq. 9, the state of the request can be examined. N is a function of sub-requests that responding to them leads to the response to the beta request. Eq. 9 indicates that an event with the dynamic and interactive nature causes the space of the request changes from space with four vectors to one vector with $\pi_{R_p+R_m+R_f+R_{io}}$ dimensional space. Changing in the state is due to the definition of N vector operators. If $N=1$, the request can be responded by a traditional RD, and if $N>1$, the request should be responded by ExaRD [12]. ExaRD is a framework for the empowerment of RD to manage events with dynamic and interactive nature. It is executed in distributed exascale computing systems by maintaining consistency with RD in traditional computing systems.

After the occurrence of an event with the dynamic and interactive nature in the process, RD redefines requests of each four resource types based on the new states of resources. To this end, one of the resources is considered as resource X (X in each redefining can be one of the four types of resources of IO, process, file, memory). The state of resource X is considered constant by RD. Resource discovery redefines the requests. This redefinition is conducted considering the impacts of the state of the requests on other resources except for resource X, as well as considering the impacts of $\pi_{R_p+R_m+R_f+R_{io}}$. Equation 9 indicates that an event with the dynamic and interactive nature for the characteristics vector of the request may influence on every four types of the resource. Thus, vectors describing characteristics of the request for each four-type resources should be redefined. Changing the state of the resource causes cross product is changed to the sum of the vectors, which is due to the definition of N vector operators on the 4-dimensional space of the characteristics of the request. This definition causes four vectors of the characteristics of the request to be changed to the responding vector to the beta request. Operators with N members cause all parts of the request to be responded. For each event with the dynamic and interactive nature, RD in distributed exascale computing systems considers it as a factor for changing the state of the requests and changing characteristics of the requests. Thus, based on Eq. 9, RD redefines characteristics of the request and the response. Similarly, Eq. 10 can be defined using Eq. 5:

$$Resource_{Vector}^{Time}{}_{D\&I} = \mathrm{T}(Resource_{Vector}) = \begin{bmatrix} \left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector}; Resource_{Vector})\right]_t & \left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector}; Resource_{Vector})\right]_t \\ \left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector}; Resource_{Vector})\right]_t & \left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector}; Resource_{Vector})\right]_t \end{bmatrix} \quad (10)$$

As can be seen in Eq. 10, after the occurrence of events with the dynamic and interactive nature, the function describing the space of the resource is changing from the cross product under the linear map T to the matrix form. Resource discovery under the occurrence of an event with the dynamic and interactive nature tries to respond to the request. Thus, it creates a matrix of capabilities of each resource based on the time vector. Matrix elements represent the empower vector of resources for every four types of resources. These vectors are in a way that the impacts of use or disuse of each type of four resources are considered. In contrast to the vector describing characteristics of the request, each empowers vector is needed separately by RD. Each vector is a function of time and may change over time.

According to Eq. 9 and Eq. 10, Eq. 5 can be rewritten based on Eq. 11. Equation 11 represents RD function after the occurrence of an event with the dynamic and interactive nature in the distributed exascale computing system. The impact of an event with the dynamic and interactive natures on RD function causes Eq. 5 to be changed to Eq. 11:

As can be seen in Eq. 11, the first part of the vector or part A indicates that the resource vector can respond to which parts of the characteristics vector of the request. An

$$f(Resource\ Discovery) =$$

$$\sum_{k=1}^{m}\left(\frac{1}{4}\sum_{n=1}^{4} i^{n}\right)\left\|\left(\overbrace{\begin{bmatrix}\left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector};Resource_{Vector})\right]_t & \left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector};Resource_{Vector})\right]_t\\ \left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector};Resource_{Vector})\right]_t & \left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector};Resource_{Vector})\right]_t\end{bmatrix}}^{A}\right)\right\| / \left|i^{n}\begin{bmatrix}\left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector};Attribute_{vector_k})\right]\end{bmatrix}\right| \tag{11}$$

$$\left\|\begin{bmatrix}\left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector};Attribute_{vector_k})\right]\end{bmatrix}\right\|^{2} * \overbrace{\begin{bmatrix}\left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector};Attribute_{vector_k})\right]\end{bmatrix}}^{B}$$

event with a dynamic and interactive nature may cause changes in the functionality of the resource, characteristics of the resource, and capabilities of the resource that can be presented to the requesting process. On the other hand, an event with a dynamic and interactive nature can influence the functionality of the process, as well as on the functionality of the global activity of which the process is part of it. The occurrence of each of the above mentioned five states (or a combination of them) is considered by part A. By because the state of the resource and the process are changing after the occurrence of an event with the dynamic and interactive nature, Part A redefines resource capabilities, necessities of the process, and the responding pattern to the request. Part A determines whether, after the occurrence of an event with the dynamic and interactive nature, the founded resource is still able to respond to the request of the process. The power of I vector represents the dimension of a vector in cubic Tesseract after the occurrence of an event with the dynamic and interactive nature.

In Eq. 11, quadratic form of

$$\left\|\begin{bmatrix}\left[\pi_{R_p+R_m+R_f+R_{io}}(Process_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(Memory_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(File_{Vector};Attribute_{vector_k})\right]\\ \oplus\left[\pi_{R_p+R_m+R_f+R_{io}}(IO_{Vector};Attribute_{vector_k})\right]\end{bmatrix}\right\|^{2}$$

indicates that the resource is influenced by the events with dynamic and interactive nature. Indeed, following the occurrence of events with the dynamic and interactive nature, the quadratic form describes how the functionality of the resource is changed in terms of different capabilities that can be presented to the process and the resource in which dimension (I/O, Memory, File, and Process) is affected by the dynamic and interactive events. This description causes changing the priority of the selection of the resource because of RD. Depending on the necessities of the requesting process, changing incapability of the resource in each dimension can decrease or increase the priority of the usage of the resource.

In Eq. 11, vector B represents a set of activities that take place in the computational element of the requester before the occurrence of an event with the dynamic and interactive nature. In distributed exascale computing systems, each process is part of global activity. Each process influences (is influenced) from (by) processes executing on the requesting computing element. Thus, vector B represents (a) a set of activities taken place in the requester before the occurrence of an event with the dynamic and interactive nature in the responding process to the beta request, and (b) a set of activities related to processes that have interaction with the responding process.

The product of vector B and A divided by the quadratic form can be examined by two approaches. The first approach examines the impacts of changes in the state of the request on the responder. This cross-product leads to modification incapability of the resource after it is used by the requesting process. If it is assumed that both the states of responding resource and the requesting process have changed, the impact of events with the dynamic and interactive nature of the process should be applied to the resource too. In the second approach, the cross product of vectors causes the state of the request is specified after the response by the computing element k.

4.2 Resource discovery failure in distributed exascale computing systems

In this section, the concept of failure of RD due to the occurrence of events with the dynamic and interactive nature in distributed exascale computing systems is discussed.

In traditional computing systems, RD is responsible to establish consistency between the request and the resource. In such computing systems, a function such as Eq. 1 can be introduced, which contains the concept of consistency. Based on Eq. 1, RD should be able to find a resource that is at 100% consistent with the request of the process. Also, from the activation of RD to the allocation of the resource to the requesting process, function 1 does not change. Thus, in contrast to RD function in distributed exascale computing systems, function 1 is not a function of time. In distributed exascale computing systems, the nature of the scientific application is in such a way that at the time of the design of the system, it is not possible to have access to all information related to the responding structure of processes [3]. To Recognize and explore the unknown parameters, a system is needed that would be able to create the responding structure during the execution of the program. This is because, at the beginning of the execution of the program, all variables participating in the global activity and all their necessities cannot be predicted.

In distributed exascale computing systems, if the computing system cannot respond to the request of the process, RD is responsible to create a responding structure outside of the local computing system [27]. Resource discovery in distributed exascale computing systems should be able to manage and control events with dynamic and interactive nature, which may have an impact on the functionality of RD. Making changes in processes that are effective on the functionality of RD increases the possibility of failure of the activities related to RD.

Because of the occurrence of events with the dynamic and interactive nature in computing processes, there is a possibility for creating changes in the space of the requesting process and the process owns the resource [12]. The space of the requesting process is defined based on limitations and constraints governed on the request, while that of the responding space is defined based on limitations and constraints governed on the responding process. To respond to the request, the variables of these two spaces should be appropriately mapped. In distributed exascale computing systems, due to the definition of the dynamic and interactive nature and the possibility of the definition of global activity and a responding structure, a computational element may not be able to respond to all parts of the request. In such a condition, the responding structure responds to the request [28]. As a result, similar to the consistency concept as the functionality of traditional computing systems, the concept of the degree of consistency between the request and the response is defined. Thus, a mapping for RD is acceptable if relative or absolute homomorphism can be created between part or all of the requests and capabilities of the discovered resource.

Resource discovery is active outside the system. Thus, in addition to factors contributing to the failure of RD including traditional factors or factors caused by the events with the dynamic and interactive nature, environmental factors also may cause the failure of activities of this unit. One of the most important environmental factors is the nonexistence of a responder. If RD finds a computing element that can respond to the request, there is no obligation that this element to provide services until the end of the usage of the resource by the requester. The other factor is the scalability of the system and its connection with other systems [30]. This is due to the implementation of resource management in a distributed form that leads the system to be dynamic. The dynamic nature of the system causes failure of RD; thus another machine should be found to respond to the request. Thus, the failure of RD can lead to several executions of activities related to RD.

During the execution of activities related to RD, events with the dynamic and interactive nature in computing processes cause functional dependency of processes to time. Thus, for the requesting process four states can be considered:

(1) After the occurrence of events with the dynamic and interactive nature in the process, the request for the process remains unchanged. In this condition, RD conducts related activities based on the previous routine. In this condition, because of RD Eq. 1 and Eq. 11 are the same. Unchanging the space of the characteristics of the request causes the constraints governed on the request do not change. In this study, only failures caused by changing the state of the requesting process during the execution of activities related to RD are considered. Thus, there is no change in the definition of the resource. As a result, Eq. 3 to Eq. 7 are equivalent to Eq. 8 to Eq. 11.
(2) After the occurrence of the dynamic and interactive nature in the process, the space of the request changes. In this condition, each or all of the time and location constraints and the type of resource may change. Besides, a request with new constraints can be created. In these two cases, even if RD can find the machine containing the resource, the machine may not be able to respond to the request of the process. If the time constraints of the request are not violated, RD should find another machine for the response to the request; otherwise, RD fails. Besides of failure of RD, the occurrence of events with the dynamic and interactive nature in the process changes the space of the request and RD function. Changing the space of the request leads to changing the space describing the request, meaning that Eqs. 8 and 9 and RD function are based on Eq. 11.

(3) Due to the dynamic and interactive nature in the process, the system is changing in a way that the local computing system can respond to the request of the process. In this condition, RD that has been activated before creating changes in the state of the system should be stopped and RD fails. This failure cannot be described by Eqs. 8 to 11. In this failure, the reason for the activation of RD is violated. In this condition, given resource management, RD is not required to be activated. Given RD in Eq. 9, the request is eliminated. Thus, in Eq. 11, RD functionality is a zero vector.

In addition to the abovementioned items, [2] defined four types of RD failure due to the occurrence of events with dynamic and interactive nature. In View influence state, constraints governed on the request are changing in a way that the request cannot be responded. In this state, the request vector is instead of the request vector described in Eq. 7, and $\text{Request}_{D\&I}$ is instead of the request vector described in Eq. 8. Thus, $\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}$ which shows the subtraction of the abovementioned vectors can be defined. If until the View Influence, RD is examined n number of computing elements and has access to n computing elements of the global activity that can respond to the beta request, $\{\text{Resource Discovery}_1, \ldots, \text{Resource Discovery}_n\}$ can be defined.

Each element of $\text{Resource Discovery}_i$ represents a set of activities that RD conducts in the computing element I to respond to the beta request. The space of resource discovery defined in Eq. 11 represents the functionality of RD after the occurrence of the dynamic and interactive nature in the process. The space of resource discovery is defined based on the scalar product of $Resource_{Vector}$ $Attribute_{vector_k}$ and $Request_{Beta}$. $\left[\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}\right]_k$ represents subtraction of the vector describing the state of the request after the occurrence of the event with dynamic and interactive nature from the same vector before the occurrence of the dynamic and interactive nature. This vector represents a set of activities and necessities that have been created or eliminated due to the occurrence of the event with dynamic and interactive nature. Resource discovery should compute this vector in n computing elements members of its global activity. Resource discovery calculates Extender based on Eq. 12. If the size of the Extender is lower than a specific amount, the failure of View Influence has occurred.

$$Extender = \sum_k(\text{Resource Discovery } |\left[\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}\right]_k).\left[\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}\right]_k \quad (12)$$

As can be seen in Eq. 12, in each computing element that is a member of the global activity for responding to the beta request, RD computes $\text{Resource Discovery} | \left[\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}\right]_k$. The resulting vector indicates after the occurrence of an event with the dynamic and interactive nature of each of the processes of the global activity, whether or not RD can respond to the beta request. Scalar product of $\left[\overrightarrow{\text{Request}_{D\&I}}-\overrightarrow{Request}\right]_k$ the mentioned vector is indicative of the impact and importance of this vector.

If RD cannot calculate the size of the Extender vector, or there is not a value for the Extender vector, the global activity for responding to the beta request is failed. This is caused by the fact that due to the occurrence of events with the dynamic and interactive nature either the request is violated or interaction of the requesting process with other processes that are members of the global activity caused violation of the global activity for responding to the beta request. Resource discovery determines the value and direction of the Extender vector. This value is determined based on the history of the conducted activities. The direction of the Extender vector is always perpendicular to the direction of the RD vector.

If the Extender vector cannot be defined, the View Influence is caused by the failure of RD. In this condition, the state of the request that is responding by RD is different from the state of the request in the requesting process. The result of the responses to the beta request should be in a way that to be different from the necessities of the beta request. In the distributed exascale computing systems, RD tries to find part of the response to the beta request in each computing element. The resulting responses may not satisfy the bet request.

In the state of the Backward Influence, due to the occurrence of events with the dynamic and interactive nature in the process, the state governed on the responding resource changes following the find of the resource and its addition to global activity. If changes in a resource or several types of resources are in a way that cannot respond to the beta request, activities related to RD are failed. Similar to the argument presented for the View Influence, Eq. 13 can be defined for the Backward Influence:

$$Extender_{resource} = \sum_k (\text{Resource Discovery} \mid \left[\overrightarrow{\text{Resource}_{D\&I}} - \overrightarrow{Resource}\right]_k). \left[\overrightarrow{\text{Resource}_{D\&I}} - \overrightarrow{Resource}\right]_{kk} \quad (13)$$

As can be seen in Eq. 13, RD in each computing element that is a member of the global activity related to responding to the beta request calculates

$$(\text{Resource Discovery} \mid \left[\overrightarrow{\text{Resource}_{D\&I}} - \overrightarrow{Resource}\right]_k)$$

. The resulting vector indicates the capability of RD for responding to the request. If by using the created and eliminated resource, RD cannot respond to the beta request after the occurrence of an event with the dynamic and interactive nature, RD is failed. The scalar product of $\left[\overrightarrow{\text{Resource}_{D\&I}} - \overrightarrow{\text{Resource}}\right]_k$ the mentioned vector indicates the impact and importance of the change that has taken place. This also obeys the rules of the Extender vector. The value and direction of the $\text{Extender}_{\text{resource}}$ is determined by resource management. If the $\text{Extender}_{\text{resource}}$ vector cannot be defined, the Backward Influence caused the failure of RD.

Conditions governed on the request may change in a way that impossibility of the usage of the discovered resource and necessity for the rediscovery of a new resource for the formed request cause inability to respond to the beta request with its constraints and limitations. In this case, RD is failed. In the View and Backward states, global activity is running. The aim of examining activities related to RD is to find the possibility of the continuation of these activities. In the Time Influence state, the activity related to RD ends, and RD allocates resources to the beta request. In this condition, the occurrence of events with the dynamic and interactive nature in the process may cause a specific resource not to be used by a chain of resources or the requesting process. Thus, the aim is to examine the possibility of re-beginning of activities related to RD considering the time constraints. In this type of failure, the aim is to find whether the time constrains governed on the request are in a way that activities related to RD can be continued. To this end, (1) $\beta = \{\alpha_1, \ldots, \alpha_n\}$ indicates activities that should be done in the requesting process to finish RD; (2) α_s indicates activities that should be occurred at the time of the occurrence of events with the dynamic and interactive nature if no changes occur in the process of RD; (3) α_r indicates the occurrence of the event at the time of the occurrence of the dynamic and interactive nature.

If matrix f exists, by considering time constraints, this matrix tries to do RD. Using Eq. 14, it is possible to decide the possibility of management of the failure due to Time Influence, as well as the occurrence of the failure of RD.

$$if\ Exist \frac{\delta f}{\delta t}[\sum_s X_s\ \alpha_s, \sum_s Y_r\ \alpha_R\ ](THEN\ Time_{influence} = Acceptable \mid \frac{\delta f}{\delta t}[\sum_s X_s\ \alpha_s, \sum_s Y_r\ \alpha_R\ ] = \frac{\delta f}{\delta t}\left[\sum_{r,s} \overline{Y_r}\, A_{rs} X_S\right] \quad (14)$$

In Eq. 14, matrix f is defined in the form of $\left[\left[\pi_{R_p+R_m+R_f+R_{io}}\left(\delta Process_{Vector}/\delta t ; \delta Resource_{Vector}/\vartheta t\right)\right]_t^{\delta t}\left[\delta File_{vector}/\delta t ; \delta Resource_{Vector}/\vartheta t\right]_t^{\delta t}\right.$ $\left[\pi_{R_p+R_m+R_f+R_{io}}\left(\delta Memory_{vector}/\delta t ; \delta Resource_{Vector}/\vartheta t\right)\right]_t^{\delta t}$ $\left.\left[\pi_{R_p+R_m+R_f+R_{io}}\left(\delta IO_{vector}/\delta t ; \delta Resource_{Vector}/\vartheta t\right)\right]_t^{\delta t}\right]$. In this matrix, the derivative of the capability of each resource (or resources) relative to time is based on consideration of the impacts of the capabilities of other resources. Matrix f represents the capabilities of each resource concerning time. Given RD, a response to the request is more important. Based on this, RD examines each resource to find which activities are possible and these activities respond to which parts of the request of the process. Thus, given RD, the definition of a resource is defined based on activities of an ordered basic member β. Thus, Eq. 14 is used to calculate matrix f.

In Eq. 14, X_s represent Eq. 3 for the event S, and Y_r represents Eq. 11 for the event R. α_s and α_r are defined based on Eqs. 3 and 11, respectively. In Eq. 14, A_{rs} represents the scalar product of two parts of A in Eqs. 3 and 11 for events S and R. In this equation, X_{rs} indicates that the vector characterizes the resource can respond to which parts of the requesting vector when the impacts of the dynamic and interactive nature in the event R are considered. This definition is based on the fact that if the S event has not occurred, which responding structure has existed, and after the R event, which parts of the responding structure can be executed.

In the Global Influence state, because of RD, changes of the global activity in which the requesting process is part of it,

cause changes in the nature of the request and constraints governed on the request. As was shown in Fig. 1, because of RD, activities related to RD are global activities that are part of the initial global activity. The requesting process is a member of this global activity. Thus, in the Global Influence state, changing part of the global activity causes the functionality of RD to be affected.

In the View Influence state, the reason for failure is that the state of the request is not the same as the necessities of the process. In this condition, due to the occurrence of events with the dynamic and interactive nature in the process, the View Influence may occur which leads to failure of RD. In the Backward Influence, the reason for failure is changing the state of the machine containing the resource. In this state, a new global activity is created after finding a resource, by which a connection between the requesting process and the resource is established. In Time Influence, global activity is stopped, and changing time constraints may fail RD. In the Global Influence, the dynamic and interactive nature does not occur in the computing element containing the requesting process or in the element containing the responding resource. Constraints and limitations governed on the procedure of RD also do not change direction due to the occurrence of events with dynamic and interactive nature. This is due to the functional nature of the process in distributed exascale computing systems. Each process is part of global activity and RD is part of the global activity in which the process is a member of it. The occurrence of events with the dynamic and interactive nature in other parts of the global activity may lead to failure of RD.

In failure caused by the Global Influence, we are trying to find whether the occurrence of an event with the dynamic and interactive nature in each part of the global activity can lead to failure of RD. To this end, each process of global activity can be considered as a vector. The size of this vector, the importance of the process during the execution of the global activity, and its direction is obtained from the sum of vectors file, I/O, Process, and Memory. In this condition, each global activity can be considered as the cross product of these vectors. To examine the impact of the occurrence of an event with the dynamic and interactive nature, in one (or more than one) vector of the global activity on the sum vector of the effective process for RD, the linear operator D&I_Mapping can be used. D&I_Mapping is a normal operator that is applied to the finite cross product of the global activity. For the occurrence of an event with a dynamic and interactive nature, this operator examines the impact of the dynamic and interactive nature of activities related to RD. To this end, it is assumed that c_1 to c_k represent values of vectors that have a dynamic and interactive nature. In this condition, W_j represents characteristics of vectors related to c_j based on the sum vector of activities related to RD. E_j represents the orthogonal projection of the Global Activity vector over W_j. If i and j are different, W_i is perpendicular to W_j and D&I_Mapping can be represented based on Eq. 15:

$$\mathrm{D\&I_{Mapping}} = \sum_{K} c_i E_i \tag{15}$$

As can be seen in Eq. 15, the linear operator D&I_Mapping is defined as the sum of the product of characteristic of vectors with the dynamic and interactive nature at the analyzer operator E_j. The D&I_Mapping operator represents the impact of vectors with the dynamic and interactive nature of the vector's constituent activities related to RD. After the occurrence of an event with the dynamic and interactive nature in global activity, this operator analyzes the states of vectors related to RD. Thus, this operator creates a mapping between the space of vectors with the dynamic and interactive nature and vectors related to RD, after which by considering constraints and limitations of the requesting process, this operator examines the states of vectors constituent RD. It can be proved that if Eq. 16 is valid, states of vectors related to RD do not fail if the impact of D&I_Mapping is being considered.

$$e_j(\mathrm{D\&I\_Mapping}) = E_j \qquad | \; e_j = \prod_{i \neq j} \left(\frac{x - c_i}{c_j - c_i} \right) \tag{16}$$

As can be seen in Eq. 16, the state of each process related to RD is the same as the state of the process before the occurrence of an event with the dynamic and interactive nature. Given RD, this means that the vector corresponding to RD is independent of W_i and thus is independent of the occurrence of an event with the dynamic and interactive nature. This independency from processes that are members of global activity implies that there is no correlation between these two vectors.

5 Evaluation

To evaluate the presented pattern for evaluation of failure due to the dynamic and interactive nature in activities related to RD in distributed exascale computing systems, peer-to-peer distributed computing systems (PMamut) [29], and the Cactus framework are used [31]. In Cactus, the characteristics of resources are changing over time. It is a multipurpose framework and an open-source for solving scientific and engineering problems that need parallel computing. Considering the variability of characteristics of resources, this framework has provided a flexible framework to be used in distributed systems. In the Cactus framework, a mechanism is used to choose resources that in case of a reduction of the performance of the system, allows changing allocation of the resources. In this framework, RD, process migration, and resource allocation are conducting their activities. The approach for the selection of the resource in this framework follows the matchmaking

algorithm, which can find the best resource. Requests of processes in this algorithm are selected based on the type of the operating system, a minimum memory, and minimum bandwidth.

In both Cactus and PMamut [29] systems, because of RD, the dynamic and interactive nature means that a situation is created that has not been considered in the executing pattern of RD. In [29], to analyze the failure of RD caused by the formation of the event with the dynamic and interactive nature, the resource manager uses the region to manage the system [26]. As a result, in this type of PMamut [29], four regions corresponding to four types of the main resource defined by the operating system are considered. As such, global activities can be defined in the PMamut [29] in which the occurrence of an event with the dynamic and interactive nature is possible. As different global activities can be defined and the possibility of changing the state of each computing element in the system, failure of RD is possible. In PMamut [29], the mechanism of ExaRD is used to respond to requests for which the local computing system cannot respond to them. Each part of the global activity (each process under execution in each computing system) can have access to each of the four types of the main resource that may not exist in the local computing system. Given the resource management, this request means the necessity to call RD.

If the matchmaking algorithm fails in the Cactus framework, another resource is allocated to the process based on the migration algorithm. In the Cactus framework, failure of activities related to RD occurs when the allocated resource to the process cannot satisfy constraints governed on the request of the process. By extension of the framework introduced in [29] for supporting events with dynamic and interactive nature, four types of failure can be considered in the execution of activities related to RD. The first one is the state in which RD cannot find the resource considering constraints and limitations governed on the request. This state is equivalent to the failure of Time influence and Global Activity influence. In the second state, as RD in the framework of Cactus tries to find the best resource for the requesting process, finding the best resource may take some time. Thus, the failure of Time influence occurs. In the third state, changes in the functionality of the resource cause failure of RD, and RD should be called again. This is equivalent to Backward influence failure. Due to the occurrence of events with the dynamic and interactive nature and changing necessities of the process during execution, failure of the View influence is also possible. Thus, failure with the type of View can be regarded as a failure due to the occurrence of an event with a dynamic and interactive nature.

To find out whether or not the pattern presented in this paper presents a logical description of the failure caused by the dynamic and interactive nature during the execution of RD, in systems [27, 30] two global activities are executed. Charm [32] and WRF [33] models need systems with high computing and processing power. Each of these models uses the computing resources of the system based on global activity. Thus, two global activities are executing in the computing system. In each time, each computing element of the system can contribute to the execution of one or two activities of the global activity.

The number of computing elements is 180 in the system. Creating a computing system using 180 computing elements makes it possible to consider the selected computing system as an extensive system for each of the two models. Generally, the abovementioned models are being executed on fewer computing elements. Thus, their execution of over 180 computing elements makes it possible to analyze the state of models when they are over an extensive system.

Considering the nature of the distributed exascale computing systems and the necessity to define the initial computing system, 70 computing elements were considered for the initial computing element. 70 computing elements are coincident with the initial necessities of the computing processes related to the scientific applications. During the execution of the models, if the computing process needs new resources to continue the execution, RD [26, 29] extend the system and adds new resources to the system.

To create events with the dynamic and interactive nature of machine 48, a specific version of the resource management is used. In RD of this system, in addition to execution of activities related to RD and creation of global activity to respond to the request of the process, based on the equations introduced in this study failure due to the formation of the dynamic and interactive nature during RD is analyzed. Based on Eq. 11, resource management [29] decides about the functionality of RD, whether the current functionality of RD leads to failure and the type of failure.

The reason for selecting machine number 48 is since this computing element participates in the global activities of both models most of the time during the execution of scientific applications. The Hardware configuration of this computing machine is equivalent to all other computing machines in the system. For each global activity, one page based on what is mentioned in [28] is considered. Most of the time during the execution of the abovementioned models, computing element number 48 is the intersection point of two planes corresponding to global activities [28]. Each other computing element can also be selected to be examined.

In the computing element number 48, resource management describes the state of the RD in each time and for each event. In this computing element, by activation of RD, the functionality of RD is described based on Eqs. 2 to 7, particularly Eq. 7. For analysis of the failure, the resource management in the computing element number 48 analyzes the state of RD and processes own both the resource and the requester. The functionality of RD is also redefined based on Eqs. 8 to

10, particularly Eq. 10. The uncertainty nature of global activities under execution in computing element 48 and the possibility of the dynamic and interactive nature in each of the global activities cause the occurrence of an event with the dynamic and interactive nature to be possible during execution of RD based on the classification presented in [2]. The occurrence of the event with the dynamic and interactive nature may lead to a stop of execution of activities related to RD. Based on Eq. 11, resource management of the processes of the system in the computing element number 48 decides whether it is not possible to execute RD due to the occurrence of an event with the dynamic and interactive nature in processes. Besides, based on Eqs. 12 to 16, resource management of the processes decides regarding the created impact on the execution of RD and the type of failure based on what is specified in [2]. Equations. 12 to 16 that are introduced in this study cause the resource management to be able to distinguish between events with the dynamic and interactive nature effective in the execution of activities related to RD, as well as the occurrence of events with the dynamic and interactive nature that lead to failure of RD. As stated earlier, Eq. 11 describes the functionality of RD. Based on this Eq., resource management can decide whether or not to execute activities for RD. Besides, by extending Eq. 11, the resource management examines how activities related to RD stop and what is the type of failure in this element.

ExaRD that is executed in computing element number 48 can manage an event with the dynamic and interactive nature caused by the creation of new processes. It is also able to manage inter-processors communications and interactions between the environment and the system. The functionality of RD in computing element number 48 is in a way that can execute requests with a dynamic and interactive nature. It is configured in a way that can respond to normal requests in traditional computing systems. Besides, it examines situations in which events with dynamic and interactive nature may disturb RD. The system and thus the computing element number 48 in distributed exascale computing systems [29] and also [26] are examined in 100 executions. To this end, first, experiments are executed in the system with the resource management of [29] and then [31]. As conditions of both experiments are similar, some results of the experiment are specified in which the resource management [29] has been executed. The results can be extended to the state for which the resource management is [31]. In Fig. 2 the numbers of RD and call for ExaRD are shown.

As can be seen in Fig. 2, in each execution of scientific applications, RD is called on average 24 times in the computing element number 48. This means that on average in the execution of scientific applications in each computing element, 24 requests are created that lead to call of RD by the load balancer due to its inability to respond to the request. Among these 24 requests, the type of 17 requests is in a way that within which event with the dynamic and interactive nature occurs during the activity of RD; thus, ExaRD should be called to respond to them. Based on Fig. 2, it can be concluded that 72% of RD calls lead to the call of ExaRD. Based on that and with the repetition of the experiment, it can be concluded that if calls of RD that lead to activation of ExaRD are 60 to 80%, the results of this study can be used. In Table 1, the correlation coefficient between the number of calls for RD and ExaRD is shown.

As can be seen in Table 1, the correlation between the number of calls for RD and ExaRD is moderate to weak. This implies that events that lead to the call of ExaRD are independent of events that lead to the calling of RD. Thus, the presented Eqs. can be extended for each exascale distributed computing system. The results of this experiment can also be extended to other computing systems. The pattern used in this study to call ExaRD is based on calling by the RD. If in activities related to RD an event with the dynamic and interactive nature occurs, RD calls ExaRD. If there exists a meaningful high relation between RD and ExaRD, ExaRD is a specific state of RD. However, in Eqs. 8 to 11 of this study, an event with the dynamic and interactive nature is considered as a factor to change the basic structure of RD from finding a resource to the creation of a responding structure. Results of Table 1 in terms of independence of the two mentioned

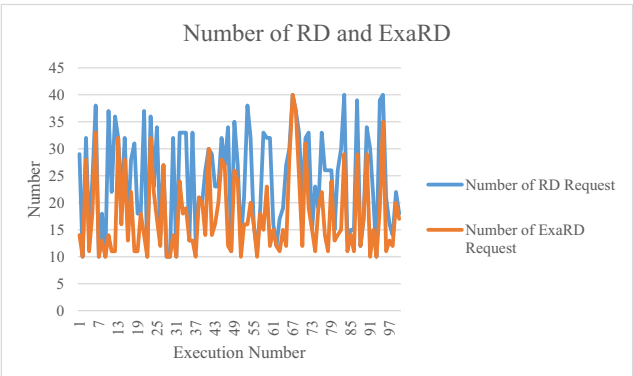

Fig. 2 The numbers of RD and call for ExaRD in distributed computing systems

Table 1 The correlation coefficient between the number of calls for RD and ExaRD

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.-6-4-8[a]	.421	.415	7.02063

[a] Predictors: (Constant), Call ExaRD

management elements cause the necessity to RD by which in case of the occurrence of activities with the dynamic and interactive nature, changes can be managed.

The number of calls for ExaRD and the number of failures related to each execution of scientific applications in distributed computing systems is shown in Fig. 2.

As can be seen in Fig. 3, on average in each execution, 14 calls of ExaRD are failed (based on a description of the functionality of ExaRD specified in Eq. 11). Given resource management, this means that during the execution of activities related to ExaRD, the dynamic and interactive nature has occurred among 14 calls of the total 18 calls. The occurrence of an event with the dynamic and interactive nature leads to failure of ExaRD. Based on what is presented in Fig. 3, in distributed computing systems [29], in 76% of the events with the dynamic and interactive nature that affect the functionality of RD, activities related to RD fail. Based on Fig. 3, if the number of execution of the experiment increases the number of events that affect RD that may lead to the failure of its activities also slightly increases. ExaRD recognizes these types of failures. On the other hand, the number of events with the dynamic and interactive nature that cause failure in activities of RD also slightly increases in case of an increase in the number of experiments. This is due to the functionality nature of RD. Other constituent elements of the resource management either remain constant or decrease with a constant slope under the increase in the number of execution of experiments. As the activity of RD is outside of the computing system, its slope is not declining. Due to the occurrence of new events in the environment of the system, either this pattern has a constant slope or its slope is increasing. As can be seen in Fig. 3, in case of a high repetition of the experiment, the occurrence of events with the dynamic and interactive nature leads to failure of RD with a slight increasing slope. By examining the type of failure of RD, it can be seen that most of the failures follow the Backward influence and the Global Activity influence. The nature of these failures is outside of the computing system. Creating changes in the state of the computing system cause this type of failure outside of the system. One of the most important reasons for changes in the number of events with the dynamic and interactive nature that lead to the failure of ExaRD implemented in the PMamut framework [29] is the pattern of RD concerning global activity and lack of maintenance of the information of the environment of the system.

Correlation between the number of events lead to the call of ExaRD and the number of failures due to the occurrence of an event with the dynamic and interactive nature during the execution of activities related to ExaRD is shown in Table 2.

As can be seen in Table 2, the correlation between the number of calls of ExaRD and failure of ExaRD is 0.468, implying a moderate to a weak correlation between the two variables. This is caused by the occurrence of events with a dynamic and interactive nature. The pattern of the dynamic and interactive nature is completely independent of distributed exascale computing systems including [29, 31]. The independency of the number of events with the dynamic and interactive nature from the number of failures of ExaRD implies that failure depends on other factors in the computing system. Table 2 indicates that only 40% of failures related to activities of RD are due to the occurrence of events with dynamic and interactive nature. The remaining 60% is due to other defined factors in the system and environment. By examining the conducted experiments, it can be concluded that the impacts of the

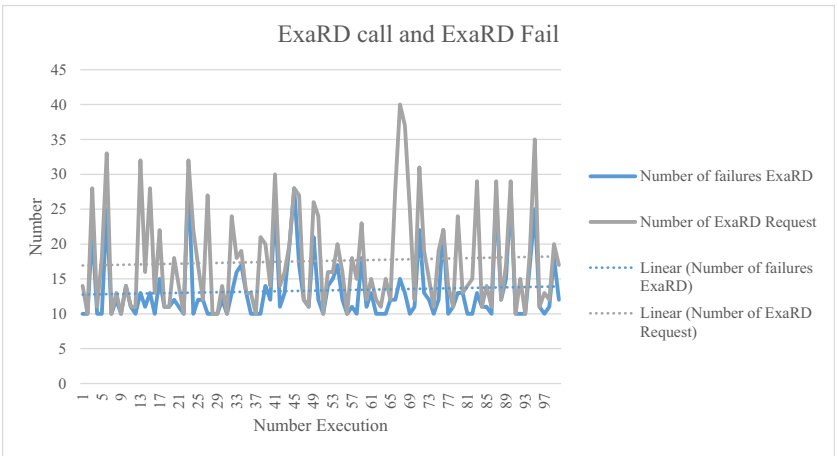

Fig. 3 The number of calls for ExaRD and the number of failures in each execution in the PMamut framework

Table 2 Correlation between the number of ExaRD and failure of ExaRD

Model summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.-6-8-4[a]	.468	.463	5.36784

[a] Predictors: (Constant), Number of ExaRD failures

system on the global activity that lead to the calling of RD, the state of the requesting process, and the impacts of the environment on the computing element are among the most important factors contributing in failure. As a result, in 40% of experiments there exists consistency between the pattern of the request that leads to activation of ExaRD and the occurrence of events with the dynamic and interactive nature that leads to failure of RD (Fig. 3). On the other hand, in the remaining 60%, due to other factors, this consistency does not exist.

In Fig. 4, the number of calls of ExaRD in which events with the dynamic and interactive nature have occurred and the number of failures of activities related to RD is shown. On average in each execution, 7 events with the dynamic and interactive nature are recognized by the Cactus framework during activities of RD. This implies that in the case of implementation of Eq. 11 in the Cactus framework, the resource management is only able to detect 7 events with the dynamic and interactive nature that affect the functionality of RD, 4 of which cause the failure of RD. Thus, in the Cactus framework, on average 52% of events that are effective on the functionality of RD cause failure of RD. ExaRD implemented in the Cactus framework can manage 50% of 7 events with the dynamic and interactive nature with which functionality of the system is changed in a way that has led to the one (or more than one) state of failure. There is no strategy for other events that are not detected.

As can be seen in Fig. 4, the number of events with the dynamic and interactive nature that affects the functionality of RD is decreasing when the number of experiments increases, implying that for the infinite number of experiments, it nearly becomes zero. Resource discovery has the required mechanisms to confront events with the dynamic and interactive nature that are discovered by it and may result in failure of RD. This is due to differences between two frameworks of [29, 31] in identifying events with the dynamic and interactive nature and thus management of these events. On the other hand, a significant difference between the slope of the number of discovered events and the number of failures due to the occurrence of events with the dynamic and interactive nature in two frameworks indicates that when the gathered information in the environment increases by the resource management, the ability of RD in discovering events with the dynamic and interactive nature increases. This means that RD not only finds resources outside the system but also creates a new responding structure outside of the system. The creation of the new responding structure implies having information about the environment and changes caused by the impact of the system on the environment and vice versa.

As can be seen in Table 3, the correlation between the number of events with the dynamic and interactive nature that leads to failure of RD and the number of calls for RD is moderate to strong. By examining Tables 2 and 3, it can be concluded that failure in activities of ExaRD and the number of calls for ExaRD in the Cactus framework has the same and related pattern in 60% of cases. However, in the framework [29], only 40% of cases have the same or related pattern. This is due to other environmental factors that lead to the failure of RD in the [29] framework.

The number of failures related to Eqs. 12 to 16 are shown in Fig. 5. On average for each execution, from all failures of RD due to the occurrence of an event with the dynamic and interactive nature, 6.68 Backward failures have occurred. By

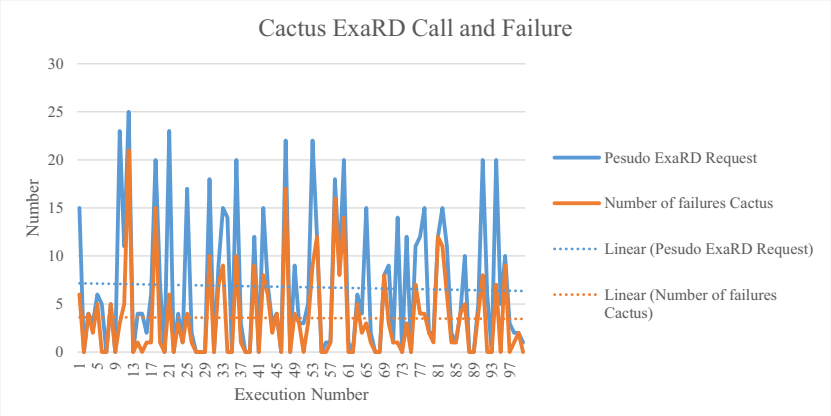

Fig. 4 The number of calls of ExaRD and failures in each execution in the Cactus framework

Table 3 Correlation between the numbers of ExaRD and failures of ExaRD in the Cactus framework

Model summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.-8-0-3[a]	.645	.641	2.68098

[a] Predictors: (Constant), Number of Cactus failure

considering the information of ExaRD in the [29] framework, it can be concluded that 50% of failures are Backward, indicating the uncertain impact of the environment on the functionality of RD. The existence of uncertain functional patterns for RD increases the possibility of the occurrence of events with dynamic and interactive nature. On the other hand, due to the absence of certain guarantees regarding the continuation of the response to the created global activities in the environment by RD, events with the dynamic and interactive nature caused by environmental changes may occur. Besides, as can be seen in Fig. 5, if the number of repetition of the experiment increases, the nature of events with the dynamic and interactive nature that lead to the Backward failure is in the way that ExaRD cannot recognize the exact pattern of the event and thus cannot prevent failure. One of the most important reasons that 50% of failures are Backward is related to the fact that events with dynamic and interactive nature are undetectable by ExaRD.

As can be seen in Fig. 5, on average in each execution, from all failures of RD caused by the occurrence of events with the dynamic and interactive nature, 3.39 of them have the type of Global Activity influence. Considering the information of ExaRD in the [29] framework, 25.39% of failures have the type of Global Activity, indicating the functionality concept of Global Activity. The occurrence of this type of failure is due to the interaction between global activity and other computing elements effective on the functionality of global activity in the system. The occurrence of 3.39 failure of the type of Global Activity indicates that on average 25% of failures are related to the occurrence of an event with the dynamic and interactive nature that occurs in the system. If the number of experiments related to the examination of Global Activity increases, the number of failures with the Global Activity slightly decreases. This is due to the capability of RD in recognizing the pattern of events with dynamic and interactive nature.

As depicted in Fig. 5, on average in each execution, from all failures of RD caused by the occurrence of events with the dynamic and interactive nature, 1.57 of them have the type of Time influence. Considering the information of ExaRD in the [29] framework, it can be concluded that 11.76% of failures have the type of Time. Thus, on average, in 12% of cases, events with the dynamic and interactive nature cause changes in the time constraints governed on the request. This change is in a way that violates the request. If the number of experiments related to Time failure increases, the slope of the number of Time failures is decreasing. This is due to the pattern of the functionality of RD and recognition of the type of requests, as well as constraints governed on different requests and areas in the [29] framework.

On average in each execution, from all failures of RD caused by the occurrence of events with the dynamic and interactive nature, 0.89 (6.6%) of them have the type of View influence (Fig. 5). When examining the system for a long time, the occurrence of this type of failure is decreasing, implying that the nature of the request has not changed during activities related to RD.

The nature of the executed scientific applications is such that after the passage of time intervals and the occurrence of specific events, Charm and WRF models are repeated. Thus, in addition to the Backward failure that occurs due to the impact of factors outside the system, the number of other types of failures is also decreasing after several repetitions.

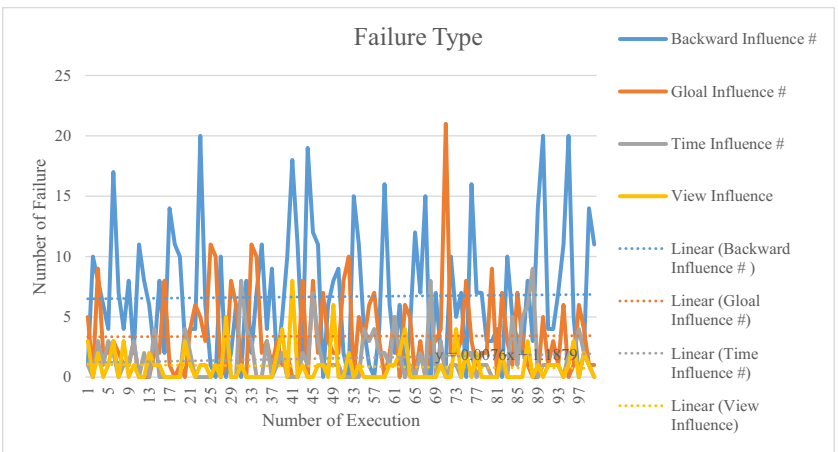

Fig. 5 The number of different failures of RD.

6 Discussion

Traditional RD and ExaRD are responsible to respond to a request by finding a resource and making it consistent with the necessities of the request. However, these two units are different in terms of the way that they are doing these tasks and the governing conditions in these two tasks. In traditional RD, finding a resource takes place without changing effective factors for finding a resource. This implies that the generator of the space of RD, elements that are influenced by RD, and elements influences on RD do not change. Considering the definition of the global activity, the global activity that has led to call RD, as well as constraints governed on this global activity, do not change during activities of RD. Thus, the failure of RD is only defined in terms of finding the requested resource. If the requested resource cannot be found, RD fails. This type of failure is due to functionality failure of RD. In functionality failure, the requested resource cannot be found in the space searching by RD. As mentioned in Eq. 1, functionality failure of activities related to traditional RD is due to inconsistency between the request and the requested resource considering the constraints of the request. Failure of traditional RD leads to non-execution of the request of the process; thus execution of the global activity stops. In this condition, constraints of the request of the process should be changed in a way that the resource management would be able to recall RD to find the resource in another environment. Eq. 1 introduced in this study describes the functionality of RD based on vector concept. Considering the state of cost vectors and the time is taken to find the requested resource, Eq. 1 can decide regarding the possibility of the failure. Generally, in computing systems, scientific applications with high reputational frequencies are executing. Repetition causes the resource management would be able to decide based on results of Eq. 1 and the history of execution of similar activities whether or not to continue the execution of activities of RD. Besides, by calculating Eq. 1, the new RD in the new environment under the remaining time and cost constraints can be examined. Thus, by examining the functionality of RD, it can be decided regarding the possibility of failure. If the possibility is not larger than a specific number, RD stops. In this way, under an acceptable cost and time interval, by the repetition of RD, the requested resource is discovered.

Given RD, adaptation means finding an element that provides all necessities of the requester. In Eq. 1, this means that adaptation between the two introduced vectors. The functionality of RD is correct if linear map T or RD can do adaptation based on Eq. 1 considering characteristics and the space of the defined request. In Eq. 1, RD should be aware of the space of the defined resources and their characteristics. This awareness in traditional systems causes the necessity for the existence of structures to gather information on resources based on history. In Eq. 1, extraction of elements comprising the space of the request based on the definition of the requesting process in the local system and the possibility of extraction of the characteristics of the request is the pivotal element for calling RD. Thus, the reason that the linear map T cannot be executed under the time and cost constraints is that RD extracts the capabilities of resources by the violation of the constraints governed on the request. In this situation, the failure that has taken place is constrained failure, which is different from functionality failure.

In ExaRD, RD is influenced by the impact of dynamic and interactive nature. The dynamic and interactive nature can influence each element that is important for finding the resource. In ExaRD, the impact of the dynamic and interactive nature causes the total adaptation to be converted to the partial adaptation. The most important consequence of this conversion is replacing a responding structure instead of finding a resource. Thus, the functional space of RD is changing from the total adaptation space to the creation of the responding structure. As a result, the occurrence of an event with the dynamic and interactive nature in global activity causes failure of RD in distributed exascale computing systems compared to traditional systems. Thus, the functional space of the RD should be redefined based on the failure caused by the occurrence of an event with a dynamic and interactive nature. The dynamic and interactive nature may influence RD but does not lead to the failure of RD. Thus, between the dynamic and interactive nature that influences RD and that leads to failure of RD should be distinguished.

The uncertainty nature of events with the dynamic and interactive nature is in a way that in each time they can influence the space generator related to RD and the spaces that are affected by them or influence them. In RD, parts of these spaces are defined outside of the management area of resource management. Thus, in addition to the dynamic and interactive nature that is effective on activities of RD inside the system, the functionality of this unit under the influence of the dynamic and interactive nature outside the system may be also affected. In RD, spaces that are effective on the activity of RD and events with dynamic and interactive nature are both defined in the inside of the system and environment. As a result, the impacts of events with the dynamic and interactive nature of the functionality of these elements and the failure of their activities are more effective. Changing RD from Eq. 5 to Eq. 11 causes RD to consider events with the dynamic and interactive nature that influence the functionality of the element. Thus, (1) changing functionality function from Eq. 5 to 11; (2) definition of the space of the request based on changing Eq. 5 to 8 and creating a responding space to the request; and (3) definition of a vector characterizing the request based on Eq. 9 make RD decide whether or not a specific event violates constraints governed on the request.

Eq. 11 introduced in this study causes RD would be able to decide how to manage the requests based on structures of the

operating system and the vectors characterizing the request, as well as based on the space of the request and the response related to the number of events with the dynamic and interactive nature that influencing the functionality of RD. Based on Eq. 11, the occurrence of events with the dynamic and interactive nature effective on the functionality of RD means the occurrence of a state that causes constraints of the request and conditions governed on the responding environment to change in a way that has not been considered in the initial responding structure. According to this Eq., based on the way that Eqs. 7 to 10 are created, the RD is informed about the number of events that lead to the specified state. To determine the type of failure, Eqs. 12 to 16 are used. As stated in the results of experiments 3 and 4, in each experiment RD implemented in the Cactus framework can discover 7 events with the dynamic and interactive nature that are effective on the functionality of RD. However, in the same experimental conditions, RD implemented in the [27] distributed system can discover 17 events with the dynamic and interactive nature that are effective on the functionality of RD. This is due to the pattern of the functionality of RD in both systems. In the implementation of RD in the [30] system those events that lead to changing the state of the responding resource to the request are considered as events that may lead to changing the functionality of RD. Changing the state of the resource is considered in three dimensions of memory, bandwidth, and the version of the operating system. On the other hand, in implementation of RD in the [27], any event that is occurred in the requesting process under the constraints governed on the request, as well as the occurrence of those activities that lead to finding a resource under conditions governed on the responding structure are considered as the effective event on the functionality of RD. As a result, RD in the [33] would be able to consider events that are effective on the request, the response, and the functionality of RD. By defining an appropriate classification for states that lead to failure, specific approaches can be used to prevent failure. In this study, Eqs. 12 to 16 are used to classify failures in 4 groups.

Mechanisms of ExaRD implemented in frameworks of [29, 31] are similar. However, these mechanisms in these frameworks discover the different number of events with the dynamic and interactive nature that may lead to failure of RD. Thus, the management of failure in a distributed exascale computing system is not completely dependent on the dynamic and interactive nature. In such systems, the dynamic and interactive nature is only partly responsible for the failure, while the pattern of management of processes in computing elements, the structure of global activity, the pattern of physical changes in the computing element, and the way that information is held are also important. The difference between the numbers of discovered events by the ExaRD in the abovementioned frameworks indicates the existence of the abovementioned factors that are not related to the dynamic and interactive nature, but influence management of the failure of RD.

As more complex factors are considered by ExaRD implemented in the [29] framework compared to that of [30] framework to detect events with the dynamic and interactive nature, the time of activity related to RD in [29] is higher than in the [31] framework. One of the most important reasons lies in the definition of functionality of RD based on states of the request and global activity, as well as the impacts of the system and environment on the activity. In the [29] framework, each activity is defined based on three indices and becomes available to RD. As a result, RD would be able to better analyze the effective events; however, the time for execution of RD increases.

The functionality nature of RD and the pattern governed on the environment causes this element would not be able to discover all dynamic and interactive events that lead to the failure of its functionality. The most important reasons lie in changing the activity of RD from the system to the environment and interaction between the system and the environment. Resource discovery is active in two different spaces in which event may form that influence functionality of this element and may lead to failure of RD. In this study, by presenting a mathematical pattern for different failures of the functionality of RD, it is discussed that events with the dynamic and interactive nature that leads to the failure might not be recognized, but based on a set of mathematical relations different forms of failures of the functionality of RD and reasons behind them can be described.

7 Conclusion

Failure of resource discovery in distributed exascale computing systems due to the difference in the functionality of resource discovery can be due to both traditional conditions and the occurrence of events with dynamic and interactive nature. As stated in this paper, it is essential to redefine the functionality function of resource discovery in distributed exascale computing systems to analyze in which state, the occurrence of events with the dynamic and interactive nature leads to failure. By introducing and re-defining the functionality function of resource discovery based on linear converters, it has become possible in this paper that resource discovery to decide based on its capability about using structures of the operating system and to decide how to define vectors of the characteristics of the request, as well as to decide about the number of events with the dynamic and interactive nature in the request and the response that are effective on the functionality of resource discovery. This makes it possible for resource discovery to decide based on the type of failure in the functionality function by the event. The results of this study

indicate that due to functional characteristics of resource discovery and execution of part of its activity outside the system, the system cannot extract all events with the dynamic and interactive nature that leads to failure of its functionality. The main advantage of the introduced mathematical function in this paper is defining different forms of failure of resource discovery. This mathematical model makes it possible to describe forms of failure and reasons for failure based on a set of mathematical rules even when the nature of dynamic and interactive events cannot be recognized.

References

1. Khethavath P et al (2013) Introducing a distributed cloud architecture with efficient resource discovery and optimal resource allocation. 2013 IEEE Ninth World Congress on Services IEEE
2. Adibi E, Khaneghah EM (2018) Challenges of resource discovery to support distributed exascale computing environment. Azerbaijan Journal of High Performance Computing 1(2):168–178
3. Mirtaheri SL, Sharifi M (2014) An efficient resource discovery framework for pure unstructured peer-to-peer systems. Computer Network 59:213–226
4. Werner H, Bornhoevd C (2020) Metadata-based general request translator for distributed computer systems. U.S. Patent No. 10,706,046
5. Dongarra J et al. (2014) Applied mathematics research for exascale computing. No. LLNL-TR-651000. Lawrence Livermore National Lab.(LLNL), Livermore,
6. Souri A, Navimipour NJ (2014) Behavioral modeling and formal verification of a resource discovery approach in grid computing. Expert Systems with Applications 41(8):3831–3849
7. Khaneghah EM, Aliev AR, Bakhishoff U, Adibi E (2018) The influence of Exascale on resource discovery and defining an indicator. Azerbaijan Journal of High Performance Computing 1(1):3–19
8. Khaneghah EM et al. (2018) Challenges of load balancing to support distributed exascale computing environment. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)
9. Messina P (2017) Update on the exascale computing project (ECP). HPC User Forum
10. Khatibi E, Sharifi M, Mirtaheri SL (2020) DPAS: a dynamic popularity-aware search mechanism for unstructured P2P systems. Peer-to-Peer Networking and Applications 13(3):825–849
11. Noghabi HB, Ismail AS, Ahmed AA, Khodaei M (2012) Optimized query forwarding for resource discovery in unstructured peer-to-peer grids. Cybernetics and Systems:687–703
12. Adibi E, Khaneghah EM (2020) ExaRD: introducing a framework for empowerment of resource discovery to support distributed exascale computing systems with high consistency. Cluster Computing:1–21
13. Saleh R, Saeidi et al (2018) A mathematical framework for managing interactive communication distortions in exascale organizations. Cogent Business & Management 5(1):1545356
14. Fahmideh M, et al. (2020) Process patterns for service oriented development. arXiv preprint arXiv:2004.09381
15. Djamaa B, Yachir A, Richardson M (2017) Hybrid CoAP-based resource discovery for the internet of things. Journal of Ambient Intelligence and Humanized Computing 8(3):357–372
16. Navimipour NJ, Milani FS (2015) A comprehensive study of the resource discovery techniques in peer-to-peer networks. Peer-to-Peer Networking and Applications 8(3):474–492
17. Harchol-Balter M, Leighton T, Lewin D (1999) Resource discovery in distributed networks. ACM, Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing
18. Navimipour NJ et al (2014) Resource discovery mechanisms in grid systems: a survey. Journal of Network and Computer Applications 41:389–410
19. Chen W (2011) Distributed device discovery framework for a network. U.S. Patent No. 7,962,605. 14 Jun
20. Trunfio P et al (2007) Peer-to-peer resource discovery in grids: models and systems. Future Generation Computer Systems Syst 23(7):864–878
21. Basu S et al (2005) Nodewiz: peer-to-peer resource discovery for grids. CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid 1. IEEE
22. Cokuslu D, Hameurlain A, Erciyes K (2010) Grid resource discovery based on centralized and hierarchical architectures. International journal for Infonomics 3(1):227–233
23. Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S (2002) Condor-G: a computation management agent for multi-institutional grids. Cluster Computing 5(3):237–246
24. Yousif A et al (2011) A taxonomy of grid resource selection mechanisms. International Journal of Grid and Distributed Computing 4(3):107–117
25. Dakkak O, Nor SA, Arif S (2016) Proposed algorithm for scheduling in computational grid using backfilling and optimization techniques. Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 8(10):133–138
26. Sharifi M, Mirtaheri SL, Khaneghah EM (2010) A dynamic framework for integrated management of all types of resources in P2P systems. Journal of Supercomputing 52(2):149–170
27. Camp DR (2006) Escape to a new dimension: a journey through space with a square, a cube, and a tesseract. Mathematics Teacher 100(3):180–183
28. Mirtaheri SL et al (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
29. Khaneghah EM (2017) 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 Apr.
30. Khaluf Y, Pinciroli C, Valentini G, Hamann H (2017) The impact of agent density on scalability in collective systems: noise-induced versus majority-based bistability. Swarm Intelligence 11(2):155–179
31. Allen G, Angulo D, Foster I, Lanfermann G, Liu C, Radke T, Seidel E, Shalf J (2001) The cactus worm: experiments with dynamic resource discovery and allocation in a grid environment. The International Journal of High Performance Computing Applications 15(4):345–358
32. Lofgren BM (2014) Simulation of atmospheric and lake conditions in the Laurentian Great Lakes region using the Coupled Hydrosphere-Atmosphere Research Model (CHARM)
33. Coniglio MC, Elmore KL, Kain JS, Weiss SJ, Xue M, Weisman ML (2010) Evaluation of WRF model output for severe weather forecasting from the 2008 NOAA hazardous weather testbed spring experiment. Weather Forecast 25(2):408–427

Elham Adibi has a Masters of Computer Science from Shahed University and currently is a member of the operating system and network laboratory at this university. She is interested in the High Performance Computing Systems (HPC) and has been involved in several related research activities over the past few years, including in resource discovery of the HPC such as grid, P2P, and exascale computing systems. Since 2015, she has been involved in a research project to improve the performance of resource discovery in distributed peer-to-peer computing systems based on mathematical mechanisms. She is also interested to design and develop a non-failure resource discovery for exascale computing systems. Elham.Adibi@Shahed.ac.ir

Ehsan Mousavi Khaneghah is a faculty member of the Computer Engineering Department of Shahed University. His research interest is the design and development of distributed computing systems. He is researching the development of a distributed Exascale computing system. He had a patent called "PMamut: runtime flexible resource management framework in a scalable distributed system based on nature of the request, demand and supply, and federalism." U.S. Patent No. 9,613,312. 4 Apr. 2017." Which proposes a framework for managing the Distributed Exascale System. His favorite research fields are an operating system, Exascale systems, parallel and distributed systems, Cluster systems, Grid systems, P2P computing systems, applied mathematics, optimization, and e-commerce. He has successful experience in running the industrial designs in high-performance computing systems. He is also a consultant of Master Plan designs in industrial areas like banks and industries, which need high-performance computing systems. Now, he is a member of the operating system and network laboratories of Shahed University. Emousavi@Shahed.ac.ir