J Grid Computing (2022) 20: 19
https://doi.org/10.1007/s10723-022-09608-z

ExaFlooding RD: A Mathematical Model to Support Unstructured Resource Discovery in Distributed Exascale Computing Environments

Zohreh Esmaeili Bidhendi · **Ehsan Mousavi Khaneghah**

Received: 3 September 2021 / Accepted: 26 April 2022 / Published online: 2 June 2022

Abstract The unstructured resource discovery has capabilities that can be used as a mechanism for creating a responsive structure and scalability in Distributed Exascale Systems for managing dynamic and interactive events. This paper introduces the ExaFlooding RD framework that can manage and control dynamic and interactive events affecting the functionality of the resource discovery in addition to activities related to unstructured resource discovery. For this purpose, while analyzing the functionality function of the unstructured resource discovery and defining the concept of request in the mentioned element, the concept of request in Distributed Exascale Systems has been redefined. Defining the functionality function of the unstructured resource discovery and redefining the concept of request as the pivotal element of the unstructured resource discovery makes it possible to define the elements of the ExaFlooding RD framework and the functionality of this framework. By using the concept of reviewing the status of executing system activities, the requesting process, and dynamic and interactive events, this framework manages resource discovery activities. Our examination indicates that this framework can manage 60% of D&I events without any dependency on the computing system performing resource discovery activities.

Z. E. Bidhendi · E. M. Khaneghah (✉)
Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran
e-mail: EMousavi@Shahed.ac.ir

Z. E. Bidhendi
e-mail: Zohreh.esmaeili@shahed.ac.ir

Keywords Unstructured resource discovery · Distributed Exascale system · Request attribute set · Dynamic and interactive events · Framework

1 Introduction

The occurrence of dynamic and interactive events in Distributed Exascale Systems may cause the system state to change in such a way that the descriptor parameters of the system describe a state that is undefined for the system manager [1–3]. Distributed Exascale systems were created to achieve 10^{18} floating Operations per the Second capability. The method of accessing these systems, unlike the concept of gigaflops to petaflops, needs to be revised in different high performance computing management systems structures as well as in making changes in the operating system of each of the elements constituting the system manager. These types of computational systems are utilized in the aforementioned systems to discover rules governing natural phenomena. In traditional distributed computing systems, distributed computational systems are used to apply rules derived from natural phenomena to reduce the time required to reach an outcome or to extend applying the rules to multiple variables. However, programs that need Distributed Exascale systems to discover the rules and extract the models governing natural phenomena, Which have not been extracted yet, include Space Weather, Brain Simulation, and Climate Simulation. Using distributed Exascale computing systems to discover the rules governing natural phenomena leads to the formation of a concept in the computational systems that is known as a dynamic and interactive event. A dynamic and interactive event occurs when due to the definition of the new variable, a new relationship between other variables with each other and with variables within and outside the system forms and as a result causes the status of the descriptive parameters of the system to be changed in a way that makes it impossible to execute the activities related to each element of the computational system manager based on the existing algorithm, and as a result, running the application fails. The occurrence of each of the three situations described constitutes the discovery of a part of the rules governing the natural phenomena that are not considered in the structure of the application and the mechanisms that are used in the structure lack the capability to manage the aforementioned situations.

In such a situation, the occurrence of a dynamic and interactive event may make it impossible to continue the activities of the system elements or execute the scientific program [4]. Numerous mechanisms and patterns have been defined to manage this situation. One of the most efficient mechanisms for managing the state of the system, after a dynamic and interactive event, is using the concept of scalability [5–9]. Any request that can be managed in the computational system by the load balancer causes the load balancer to redistribute the load based on a new recognition of the status of the available resources in the local computing system. In distributed computing systems, the load balancer has to find resources in the local computing system and make it possible to respond to the requests using local resources. This necessitates creating a pseudo-structure that carries out discovering resources and updating the status of the available resources in the computational system that is generated by the load balancer. If the request cannot be handled by the resources in the local computing system, then the resource discovery expands the system by considering new computational elements that can respond to the requests in the computational system. With this concept in mind, when the system cannot respond to its own process's request, both elements of resource discovery and the load balancer will engage in conceptual expansion of the system that is responding to the request (or requests) of the process. In distributed computational systems, in addition to the two elements discussed, running the program as a remote computational unit also can expand the system if there is information about the capability of the remote computing unit to respond to the request (or requests of the process). If the process of another resource can handle the requirements of another process in accessing a resource or executing the activities, then the mechanism of interaction and communications between processes such as distributed memory and messaging can be considered as tools for expanding the distributed computational system. In this paper, it is assumed that the set of activities performed to execute the process's request (or requests) outside the locally distributed computational system is executed, managed by the resource discovery.

Conventionally in distributed computing systems, the task of system scalability is the responsibility of the resource discovery [10]. Based on the scalability mechanism, the system manager by using the resource discovery must be able to manage dynamic and interactive events by creating a response structure at runtime. One of the most important challenges in using the concept of scalability to manage dynamic and interactive events is the lack of accurate information about the requirements of dynamic and interactive events, and how to create a response structure for them [11].

If the system manager in Distributed Exascale Systems uses scalability, it should create a response structure in which many parameters of the request, the constraints governing the request, as well as the executive elements affecting the request, it is not definite. In this situation, the system management element, unlike the requests activating traditional resource discovery element, the constraints of the request, and the elements affecting the request, is specific [12], while in Distributed Exascale Systems, the resource discovery is called parametrically by requests activating it. In Distributed Exascale Systems, due to the uncertainty of the request parameters, it may not be possible to find the resources accurately and according to the characteristics of the request. This makes the concept of similarity between the requested resource and the discovered resource in this type of computing system. This is while in the functionality of the traditional resource discovery element, either the requested resource by the process exists or the requested resource by the process is not found by the resource discovery.

In traditional computing systems, the purpose of implementing resource discovery is to find a resource.

In the distributed Exascale system, the purpose of implementing the resource discovery, in addition to the mentioned purpose, is to create a response structure to manage dynamic and interactive events. The occurrence of dynamic and interactive events may cause the resource discovery to create new response structures in the system.

One of the mechanisms used for resource discovery in traditional computing systems is the use of flood mechanisms when there is no accurate information about the response structure to the request [13, 14]. In the flooding mechanism, the resource discovery, based on the request received from the process, sends information to the neighboring computational u and repeats the sending pattern in each neighboring computational unit. This continues until either the time constraints governing the response to the request are violated, the resource discovery is unable to find the requested resource or the requested resource is discovered [15–17].

The general pattern governing the flooding resource discovery mechanism is to find the requested resource blindly [18]. This mechanism, unlike other mechanisms in which there is either a specific structure to respond to the request of the process or a specific range, is searched for the request of the process, in a wider scope and without considering any constraints on how to find the resource, the structure of finding the resource, and the limitations of the search space, try to find the requested resource [19]. Although increasing the search space, enhances the number of sent requests compared to the random walk mechanism, due to the large area used to find the resource, the probability of finding the requested resource is increased. The high extent of the area used to discover the resource makes it possible to find a resource similar to the requested resource because it has a higher number of responsive elements than other mechanisms.

In Distributed Exascale Systems, in terms of resource discovery, a dynamic and interactive event is such as a request that actives the resource discovery. In Distributed Exascale Systems, due to the lack of definition of the request activating the resource discovery, as well as the impossibility of performing system activities, if the resource discovery activities are not performed, and the possibility of defining the request with similarity coefficient has existed, the flooding mechanism can be considered as a mechanism for scalability of the system to manage dynamic and interactive events.

In this paper, while examining the concept of occurrence of the dynamic and interactive event in Distributed Exascale Systems, and its impact on the functionality of the flooding resource discovery mechanism, we present a developed flooding resource discovery mechanism that will be able to manage dynamic and interactive events. In this paper, to introduce the flooding resource discovery mechanism used in Distributed Exascale Systems, the functionality function of the flooding resource discovery is introduced, which includes the mathematical function that according to it, the flooding resource discovery operates. To enable the functionality function of the flooding resource discovery for using it in Distributed Exascale Systems, the effect of dynamic and interactive events on the functionality function of the flooding resource discovery will be analyzed. This analysis makes it possible to extract the functionality pattern of the flooding resource discovery used in Distributed Exascale Systems. This functionality pattern should be able to respond to requests in which some aspects of the request are not clear. This requires redefining the concept of responsive request by the flooding resource discovery in Distributed Exascale Systems.

2 Basic Concepts

In this section, while examining the functionality of the flooding resource discovery, and the concept of occurrence of the dynamic and interactive events in Distributed Exascale Systems, the effect of these events on the functionality of the flooding resource discovery will be analyzed. The concept of request, as the pivotal element that calls the resource discovery in a traditional computing system and Distributed Exascale Systems, will also be examined.

2.1 Flooding RD Function

In the flooding resource discovery mechanism, a request is generated in the system that has no information about what element in the system is capable to respond to it. This request has either a time limit, a location limit, or both, or none of them. The time limit is how long it takes to respond to a request. The concept of TTL is commonly used in traditional computing systems to implement time constraints. The location constraint indicates in which part of the system, the request should be answered. The functionality of the flooding resource discovery is such that it may respond to a request in another computing system.

The ability to expand in distributed computational systems is considered as the main parameter in the field of capability and functionality of these computational systems. The nature of being distributed of these systems forces the system to keep the concept of scalability, communication with other systems, as well as transparency. The distributed nature of the aforementioned computational system enables the computational system to respond to the request of the process based on either one of the expansion mechanisms that is used in this paper (mechanism of unstructured flooding scalability), or by changing the limits of the computational system, or by adding a responsive computing element temporarily to the computational system. These enable the system to execute the process without any disruption. Considering the scalability concept that was discussed above as one of the evaluation parameters in distributed computational systems can be evaluated when it leads to the successful execution of the processes in a system that is the inability to respond.

The definition of the request in the mechanism used by the flooding resource discovery is based on the attributes of the requested resource. In this mechanism, there is no precise information about which element is capable of responding to the request, so instead of explicitly expressing the request, the structure of responding to the request, and which resource responds to the request, the attributes of the requested resource is used for finding the responder element. In systems that use a structured pattern for the functionality of the resource discovery, the response to any specific request is predetermined [20]. Therefore, the resource discovery has information about which elements are capable of responding to the request. This allows the resource discovery to use the request expression pattern explicitly. This is while in the flooding resource discovery mechanism, the requesting process instead of expressing three attributes <requested resource, structure of responding to the request, responding element to the request> describes one (or more) resource attributes of the requested resource.

Failure to use the triple resource discovery space of the request that is expressed in the structured pattern makes it possible to consider a functionality function for the flooding resource discovery in the form expressed in Formula 1.

$$\begin{aligned}&\text{Flooding RD}:\\&:\ll \overbrace{\text{Process}_{\text{Requirement}}, \text{Resource Attribute Set}}^{\textit{spaces definer}}\\&>, \overbrace{\text{Query Number}}^{\textit{attribute}},\\&< \overbrace{\text{Matching},\ \text{Permission}, \text{Allocation}}^{\textit{avilable operation}} >> \qquad (1)\end{aligned}$$

As can be seen in Formula 1, the functionality function space of the flooding resource discovery is defined based on the two space requirements of the process, and the description of the resource attributes that can respond to the request of the process. The functionality function space of the flooding resource discovery function is also defined based on the request number attribute, and matching operators, permissions, and allocation. In general, the functionality function of the resource discovery is defined based on the two spaces Resource Attribute Set (*RAS*) and Request Nature Set (RNS) [21].

As can be seen in Formula 1, in the functionality function of the flooding resource discovery, the focus of resource discovery activities is based on RAS. The flooding resource discovery does not use the nature of the request properties to find the requested resource. As can be seen in Formula 1, only information about the request number attribute is considered if the number of simultaneous requests is more than one. Considering the matching operator, it is since none of the three attributes <requested resource, structure of responding to the request, responding element to the request> are specified at the beginning of the execution of the flooding resource discovery. This makes it possible, due to the execution of flooding resource discovery activities, based on the description of the resource, defining the execution of activities related to the flooding resource discovery, based on finding a specific resource in a specific responder is impossible.

The use of the matching operator in the flooding resource discovery makes the system manager find resources that are similar to the RAS, as a result of the resource discovery activities. In computing systems that use the flooding resource discovery, matching operator area, limitations, and constraints can be defined that make the response space to the request. These limitations and constraints indicate what resources, with how many similarities to RAS, can be defined as the members of the response space to the request. In resource discovery, in general, the concept of similarity is not defined. The reason for this is due to the definition of the function of the resource discovery in general based on the two spaces RAS and RNS. As the amount of resource discovery information about the RAS space increases, the response space to the request becomes more limited.

The permission operator in the flooding resource discovery is such as the permission operator in the public resource discovery. In the public resource discovery, after finding the requested resource by the process, the resource discovery obtains permissions of the required resource of the process from the local computing system manager. In the non-flooding resource discovery, a response structure is normally created when the requested resource is found, which allows the discovered resource to be used by the requesting process. Receiving permissions of the discovered resource is creating a response structure. In the case of the flooding resource discovery, sending a request to more than one neighbor may create more than one response structure. This is since more than one resource with the same RAS is found for the sent requests, and due to the delays of the sublayer of the network, more than one response for the requesting process is received.

In the non-flooding resource discovery, when the permissions of the discovered resource are obtained, due to the creation of a continuous and uninterrupted response structure, the allocation operator is executed and the discovered resource is allocated to the requesting process. In the flooding resource discovery, the result of resource discovery activities may lead to finding more than one resource about which the RAS description is true. This allows the requesting process manager to decide which resources can respond to the requesting process after discovering the resource. The result of this decision may be the same as the discovered resource space by the flooding resource discovery, or it may be part of it. After determining which resources are acceptable for the requesting process, the flooding resource discovery should implement the allocation for those resources and create a response structure for the use of that resource.

T Minimal polynomials can be considered for the RAS space associated with the flooding resource discovery. In each constituent element of these minimal polynomials such as αT^n, T represents a requested resource attribute, α is the significance coefficient of requested resource for the requesting process, and n represents the maximum detectable resources by the flooding resource discovery. In the terms of the flooding resource discovery, for each computational unit that is examined by the flooding resource discovery, T minimal polynomials are also considered. In each element of the minimal polynomials $\beta \ddot{T}^m$, $\ddot{T}$ represents a resource attribute, β represents the resource's ability to respond to requests, and *m* represents the number of resources that have the attribute of $\ddot{T}$ in the computational element. The method of defining T and $\ddot{T}$ can be expressed based on the pattern of resource definition by the operating system. In this way, for each RAS request, a minimal polynomial $f(RAS) = \alpha_1(IO)^{n_1} + \alpha_2(File)^{n_2} + \alpha_3(Memory)^{n_3} + \alpha_4(Process)^{n_4}$ can be stated. Similarly, for each computational element that can be examined by the flooding resource discovery, a minimal polynomial $f(Machine) = \beta_1(IO)^{m_1} + \beta_2(File)^{m_2} + \beta_3(Memory)^{m_3} + \beta_4(Process)^{m_4}$ can be used.

Any other development for T and $\ddot{T}$ can be considered, indicating the defined resources in the RAS and the computational elements that are examined by the flooding resource discovery. The activity of the flooding resource discovery can be considered in the form of a *W* linear operator on the *Q* finite-dimensional vector space. The *Q* vector is defined based on the *Q* system. The resource requesting process described in the RAS announces the ending of the resource discovery activity after a time constraint or any other constraint is established. The *Q* system contains all computational elements which have a resource equivalent to the resource described in RAS.

For the *Q* finite dimension vector, ordered basis such as $\{q_1, \ldots, q_n\}$ can be considered.

On this ordered basis, q_i is a number between zero and one that is never equal to zero and indicates the similarity coefficient of the resource in the computational element *i* to the resource described in RAS. Formula 2 is used to calculate q_i.

$$Wq_i = \sum_{i=1}^{n} \left(A_{ji} q_i \right) \tag{2}$$

In Formula 2, matrix A is considered as a matrix that describes the *W* linear operator on the order basis of the *Q* finite dimension vector. Each matrix element in the form of A_{ji} represents the ability to respond to the resource of type j described in the RAS polynomial, in the computational element i, in terms of the flooding resource discovery. The value of A_{ji} is the resultant of all features in the computational element i for resource j, that will be available for the flooding resource discovery. This value includes the ability of access to resource permissions, resource allocation time, concurrent use of the resource by local and global processes, as well as the ability for reusing the resource and creating a response structure. According to Formula 2, if the minimal polynomials $f(Machine)$ is divisible by the RAS minimal polynomials, then because $W(T) = 0$, the Machine computational element has the ability for responding to the request described in RAS. If the minimal polynomial $f(Machine)$ is divisible by RAS polynomials, the W linear operator represents the functionality function of the flooding resource discovery.

2.2 Flooding RD Framework

As stated in Formula 1, the functionality function of the flooding resource discovery is defined based on the RAS and the process requirements activating resource discovery. By considering the concept of Q system and the definition of Q system based on time independent variable, the framework of the functionality function of the flooding resource discovery or W can be expressed in the form shown in Fig. 1.

As can be seen in Fig. 1, the centrality of the functionality function of the flooding resource discovery is based on the two concepts of RAS and the matching operator. In Fig. 1, the space of the Q system is a function of the time independent variable whose elements are the output of the matching operator. In this type of system, the status of the computational element in which the requesting process is present does not change in a way that influences the process that actives the resource discovery. The absence of such changes makes it possible to keep the RAS space constant during the execution of the resource discovery activities. RAS space in traditional computing systems represents a detailed description of the requested resources, which in the most general situation is in the form of (Resource, Value).

As can be seen in Fig. 1, in traditional computing systems, the process requirement space contains the quantitative number of resources required by the process. Numerous patterns can be used to classify the resources required by the process. One pattern is to use operating system classification to manage resources. According to this classification, the process requirements space consists of four sub-spaces: input/output, file, memory, and process [22]. In traditional computing systems, due to the equality of the RAS space with the process request activating the resource discovery and the matching of these two spaces with each other, the RAS and Process Requirement spaces can be considered the same. The process requirements space makes the two RAS and Matching Function spaces to be in relation. The interaction of the two RAS spaces and the Matching Function with each other makes the flooding resource discovery can be able to reduce the elements of the Q space and based on the W functionality function, find computational elements that can meet the requirements of the request. As can be seen in Fig. 1, the RAS space implicitly contains time and location limitations governing the process request.

As can be seen in Fig. 1, the permissions function and the allocation method refer to how the resource is used and the constraints governing the use of the resources. In traditional computing systems, regarding the fact that

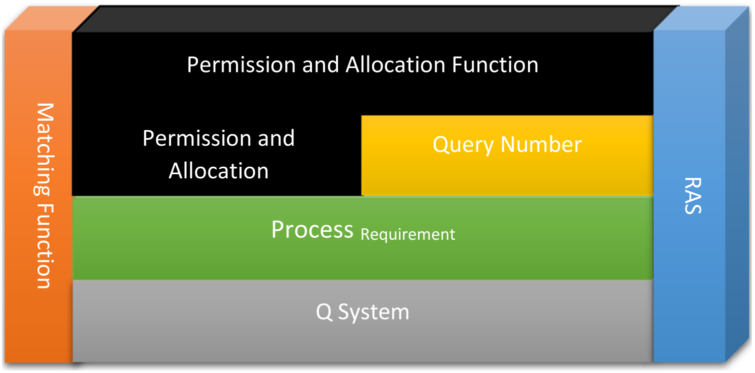

Fig. 1 Framework of functionality function of the flooding resource discovery

the resource, the constraints governing the resource as well as the status of the parameters affecting the resource are constant in the computational element, so the permissions function and how the resource is allocated are also fixed. In traditional computing systems, either the computational element containing the resource shares the resource or removes it from the sharing. In traditional computing systems, a situation other than the two mentioned about the resource and the constraints governing the resource cannot be considered. This makes it possible that resources be available for requesting process in traditional computing systems if Formula 2 is established. In traditional computing systems, RAS spaces, process requirements, request number, permissions, and resource allocation as well as matching operators are not defined as variables of time independent variables, and only Q space is defined based on time independent variables. The reason for defining the Q space based on the time independent variable is adding or subtracting computational elements that can respond to the request.

The Q system indicates the ability of unstructured flooding resource discovery to responds to the requests of the process which the local computational system cannot respond to them. The Q system is capable of handling a higher number of processes that the local computational system is not able to handle. The resource discovery and consequently the concept of scalability provide proper efficiency in the functionality of the computational distributed system. The number of processes that cannot be handled functionality the local computing system and cannot be managed by the Q system indicates that the unstructured flooding resource discovery and the concept of scalability are not able to respond to the requirements of the distributed computational system and are considered as negative points in scalability functionality. The concept of scalability in distributed computational systems is considered as a tool to increase the unstructured flooding resource discovery in the successful execution of computational processes in comparison with systems without scalability capability. Although several indicators can be proposed to evaluate the concept of scalability, the real goal of scalability by the unstructured flooding resource discovery is to prevent the failure of the program execution process in case of a lack of ability to meet the requirements of the computational process.

3 Related Work

Traditional unstructured peer-to-peer computing systems use flooding and random walk mechanisms for resource discovery operations, which result in many requests being propagated among the computational elements in the system. This causes high traffic and latency to the system, which uses time and location limitations to solve such problems. For this reason, these systems do not provide any guarantee for finding the desired resource, so the desired resource may not be found if it was present in the system [23].

Two main methods have been proposed for the flooding mechanism: 1- Pure Flooding, 2- Flood with source routing. In the first method, the computational element first checks its cache to find the desired resource If it finds the resource, it allocates that to the computing process of requesting resource; otherwise, the computational element sends the query to all neighbors for finding the resource. The second method works the same as the first method, the difference is that each computational element that receives the query puts its address in the list of addresses of the computational elements that have already received the query, and sends it for other computational elements. When the computational element containing the resource, receives the query, it uses the same list to respond to the requesting computational element [24, 25].

The task of the resource discovery, called by the requesting computing element, is to find a resource outside of the system. The resource discovery must be able to solve the challenges of occurrence the dynamic and interactive events in the requesting process. To manage and control challenges, this element must be able to expand its main activity from finding a resource to creating a response structure. The executive activities of the resource discovery will fail when there is no management and control over the occurrence of dynamic and interactive events [2, 26]. By examining and describing the concept of dynamic and interactive events in Distributed Exascale Systems, it tries to show the effects of the occurrence of dynamic and interactive events on computational elements of requesting resources.

In [21] two hybrid algorithms for unstructured P2P networks are proposed. The first is a combination of Flooding and Random Walk approaches, while the second combines Flooding with Random Walk with Neighbor's Table. The performance of each algorithm was investigated and Simulation results showed that hybrid algorithms provide the most balanced performance regarding the average number of hops, average search time, and the number of failures when compared to the basic resource discovery algorithms.

[22] evaluates classic flooding, random walk, and gossip-based resource discovery algorithms under mobile peer-to-peer (MP2P) networks and studies their performance. They compare the performance in terms of success rate, query response time, network overhead, battery power consumed, overall dropped packets, network bandwidth, packet delivery ratio, network routing load, and end-to-end delay. The result shows the performance of the flooding method is better than all the other techniques in each term.

In [23] a probabilistic flooding strategy is proposed, which manages to stop redundant flooding paths, improve flooding performance, and minimize unnecessary costs due to redundant messages. In this strategy, the decision of a node to propagate a message (or not) is based on both the popularity of resources and the hop distance from the node that initiated the query.

In [21] a framework is provided for finding resources by ExaRD, that can manage the dynamic and interactive events generated by the request created in the system. The occurrence of dynamic and interactive events in computing systems makes changes on the resource discovery by increasing the frequency of the resource discovery recalls and the effects associated with RNS and Request imaging, but in traditional computing systems if the Alpha process in responding to Beta request causes resource discovery recalls, RNS of Alpha process will not change. To reduce the changes and effects of dynamic and interactive events and respond to them, this framework changes the main activities of the resource discovery from request to $Request_{state}$ and $Process_{state}$. This change makes the resource discovery to be able to respond to 50% of these requests generated in the system by using the information collected from the status of the process and the request when dynamic and interactive events occur.

Basukoski et al. [27], provides a system that combines search techniques in unstructured peer-to-peer systems (such as the broadcast of the request) and the use of Distributed Hash Tables in grid computing systems to support resource search statically and dynamically. Utilizing dynamic queries in peer-to-peer system infrastructure based on DHT makes latency reduction of the search process and increases the efficiency of the resource search algorithm by controlling the number of messages generated per resource discovery operation.

Multiscale computing patterns are considered as a means to achieve some cases such as load balancing, fault tolerance, and awareness of the amount of energy resulting from multiscale computations with high performance. [28] discusses how multi-scale computing systems can be used to take full advantage of existing Petascale systems and to make Exascale systems more efficient.

As data volumes increase over time, their analysis is switching from single-domain networks to multi-domain and geo-distributed networks. In [29] a framework called Unicorn (unified resource orchestration framework) is provided that uses a protocol called ALTO (Application-layer traffic optimization) to support multi-domain networks while protecting the privacy of users and the proper use of various resources It also can predict the performance required to do data analysis.

In large-scale computing systems, one of the most challenging and complex issues for resource sharing and optimal use is resource management. Regarding the requirements of large-scale computing systems, in [30] an architecture called Elcore is presented. This architecture includes a set of modules that can provide capabilities such as auto-scaling, dynamic resource management, and allocation with high accuracy and multidimensional mapping between processes and resources.

As can be seen in the related works, in most unstructured flooding resource discovery, the centrality of making the resource discovery is based on two concepts of how to explore the resource according to the use of the non-structured concept and to reduce the necessary activities for the unstructured flooding resource discovery.

4 ExaFlooding RD

4.1 Request Concept According to Flooding RD in Distributed Exascale Systems

In traditional computing systems, the request leading to the call of the resource discovery can be either in the form of a request for access to a specific resource or in the form of a request for access to a resource describing its attributes. In traditional computing systems, the constraint governing the request leading to the call of the resource discovery is typically time-limitation. In traditional computing systems, the time constraint governing the request is either explicitly specified by the requesting process or by the number of times the request is published in the computing system environment.

This paper is focused on the functionality of the unstructured flooding resource discovery, the concept of scalability is used in two dimensions of increasing the probability of responding to computational processing requests, and also making a responsive structure to prevent the failure of the executing program when dynamic and interactive events occur.

In the mechanisms used for flooding resource discovery, requests are usually made in the form of describing a request and taking into account the number of times the request is published. This makes it possible to describe the concept of request in the flooding resource discovery based on Formula 3.

$$Request \overbrace{::}^{Define} \; < \overbrace{< RAS >}^{space\ define}, \overbrace{Time_{Limitation}}^{attribute}, \; < \overbrace{Answer, Similar}^{available\ operation} >> \qquad (3)$$

As can be seen in Formula 3, the concept of request is defined in the flooding resource discovery based on the RAS space. Determine what are the attributes of the requested resource? The request is described by the RAS space from the view of the flooding resource discovery based on what attributes can be described. In Formula 3, the more accurately the attributes of the requested resource are described, the more limited resource (or resources) discovered by the flooding resource discovery. Conventionally in traditional computing systems, the resource research process is initiated and performed by the flooding resource discovery based on the description of the requested resource features in general. The definition of a request based on its attributes in traditional computing systems based on flooding resource discovery is such that the description does not change during the performing the resource discovery activities.

In traditional computing systems, in Formula 3, the time constraint space normally indicates the number of times that the request is published. Determining the number of times that the request is published as a time constraint, in addition to specifying the acceptable time to respond to the request, also implicitly includes the research space of the flooding resource discovery. Due to this issue, Formula 4 can be considered as the formula of the equation of time constraint and propagation governing the element of flooding resource discovery.

$$\left[\left[\left(\frac{\delta}{\delta t}\right)-c\left(\frac{\delta}{\delta h}\right)\right]*\left[\left(\frac{\delta}{\delta t}\right)+c\left(\frac{\delta}{\delta h}\right)\right]\mu\right]=0 \qquad (4)$$

As can be seen in Formula 4, the one-dimensional wave equation relative to the requesting and activating element of the flooding resource discovery can be used to describe the equation of time constraint and propagation. In Formula 4, the variable t indicates the time elapsed for the execution of resource discovery activities by the flooding resource discovery. So δ / δt indicates how the flooding resource discovery is relative to the independent variable t at any given point in time. The variable h indicates the number of hubs passed by the flooding resource discovery. The nature of the flooding resource discovery is such that it can be considered for those uniform radius circles, each circle representing a set of hubs that result in a decrease of one TTL of request. The δ / δh variable represents the activities performed on each computational element that has been examined by the flooding resource discovery to respond to the request. In Formula 4, coefficient c indicates the effect of network parameters on the performance of the flooding resource discovery. Although coefficient c is different at any given time from the process of implementing the flooding resource discovery, it can be considered moderate. The variable μ indicates the importance of the request. In Formula 4, the variable μ can be considered a fixed value determined by the requesting process. In some of the mechanisms defined for the flooding resource discovery, the value of the variable μ can be changed according to the variable h or even the variable t.

In Formula 3, operations that can be defined based on RAS space and time constraints are two activities of response and similarity. Response activity indicates finding the resource described based on the RAS space in each computational element under consideration. According to Formula 3, the flooding resource discovery, when examining a computational element, either the element containing the resource is described or lacks the resource to be described. If the computational element contains the resource described based on RAS, then the flooding resource discovery sends the resource information as well as the response capability to the requesting process. In the absence of the resource described by the RAS, redistribution activity takes place.

The flooding resource discovery considers each process request in the form $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$ The Req_i element represents the descriptive attributes of RAS for resource i. The flooding resource discovery, in the specified computational element k, must evaluate the capability of the computational element for each of the components of the matrix, $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$. For this purpose, the flooding resource discovery uses formula 5.

$$\begin{bmatrix} \left[k_{A_p} = \dfrac{\delta(A_1,\ldots,A_n)}{\delta\left(Req_{p_1},\ldots,Req_{p_m}\right)} \right] * \\ \left[k_{A_f} = \dfrac{\delta(A_1,\ldots,A_n)}{\delta\left(Req_{f_1},\ldots,Req_{f_m}\right)} \right] * \\ \left[k_{A_m} = \dfrac{\delta(A_1,\ldots,A_n)}{\delta\left(Req_{m_1},\ldots,Req_{m_m}\right)} \right] * \\ \left[k_{A_{IO}} = \dfrac{\delta(A_1,\ldots,A_n)}{\delta\left(Req_{IO_1},\ldots,Req_{IO_m}\right)} \right] \end{bmatrix} \tag{5}$$

As can be seen in Formula 5, for each resource A member of the computational system k, the set of attributes describing the resource (A_1, …, A_n) can be expressed. In this regard, the flooding resource discovery also uses the pattern to describe the requested resource attributes by the computational process to describe the properties of each resource in the computational element k. From the point of view of the flooding resource discovery, any Req_i request that i can be a member of the set {p, m, f, IO} can be described based on the m attributes.

In Formula 5, four separate partial derivatives describe resource A for each member of the matrix $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$.If in the described request based on the matrix $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$, the value of a derivative is equal to zero, then the derivative corresponding to the derivative is not calculated. Descriptive resource derivative A, relative to each matrix element $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$ Indicates the rate of change of resource properties relative to the described properties of each matrix element $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$ Is. If the value of each k_{A_i} is a positive value, then the described source A can respond to the described Req_i request and it is possible to use this source in subsequent response structures if the source status does not change. If the value of k_{A_i} is zero, then the described source A can respond to the described request Reqi, but this source can not be used for subsequent response structures if the source status is not changed. If the value of k_{A_i} is a negative value, then the described resource A is not able to respond to the described Req_i request. In Formula 5, each of the matrix components $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$ maybe equal to zero. Its value is equal to zero. When in the matrix $\begin{bmatrix} Req_p & Req_f \\ Req_m & Req_{IO} \end{bmatrix}$, there is a zero value, then the derivative of the description of the resource properties relative to that knowledge will not be defined. According to Formula 5, the computational element may correspond to one or more of a type of resource capable of responding to a Req_i request and corresponding to one or more of a type of resource not capable of responding to a Req_i request in computing systems. Traditionally, from the point of view of the flooding resource discovery, the source lacks accountability. In Formula 5, it is assumed that each computational element can respond to more than one type of resource. If each computational element is only able to respond to one type of resource, then for the type of resource that lacks capability, the corresponding derivative is not equal to the type of resource.

So in traditional computing systems if the condition

$$\begin{bmatrix} \left[\left(k_{A_p}\geq 0\right)\right] \ and \\ \left[\left(k_{A_m}\geq 0\right)\right] \ and \\ \left[\left(k_{A_f}\geq 0\right)\right] \ and \\ \left[\left(k_{A_{IO}}\geq 0\right)\right)\right] \end{bmatrix}$$

In this case, the resource can respond to the request. If the computational element k has more than one shared resource, then for each computational resource, the resource is described and formula 5 is calculated. Formula 5 is used for the response activity. From the view of the flooding resource discovery, in terms of response activity, each resource described A, condition $\begin{bmatrix} \left[\left(k_{A_p}\geq 0\right)\right] \ and \\ \left[\left(k_{A_m}\geq 0\right)\right] \ and \\ \left[\left(k_{A_f}\geq 0\right)\right] \ and \\ \left[\left(k_{A_{IO}}\geq 0\right)\right. \end{bmatrix})]]$ is either established and the resource can respond or the condition is not met and the resource cannot answer. In Formula 3, the similarity activity is derived from the resource description based on the RAS request. The flooding resource discovery uses Formula 6 to investigate similarity activity, where a rewrite of Formula 5 is based on calculating the partial derivative of each described attribute of resource A relative to each attribute of the requested resource type.

$$k_{A_i} = \left[\frac{\delta(A_1,\ldots,A_n)}{\delta\left(Req_{i_1},\ldots,Req_{i_m}\right)}\right] = \begin{bmatrix} \delta A_1/\delta Req_{i_1} & \cdots & \delta A_1/\delta Req_{i_m} \\ \vdots \quad \vdots & & \vdots \\ \delta A_n/\delta Req_{i_1} & \cdots & \delta A_n/\delta Req_{i_m} \end{bmatrix} \tag{6}$$

In Formula 6, the value of $\delta A_j/\delta Req_{i_j}$ follows the rules stated for k_{A_i}, and a value greater than or equal to zero means $\delta A_j/\delta Req_{i_j}$ means the ability of the resource to respond to the Req_{i_j} request. A_j. In the similarity activity, there is no requirement that all m*n be a partial derivative equal to or greater than zero. When the flooding resource discovery uses the similarity activity to discover the resource, it can define a similarity percentage for each request. The percentage of similarity is equal to the number $\delta A_j/\delta Req_{i_j}$ greater than or equal to zero divided by m*n multiplied by 100. The flooding resource discovery can specify that for each Req_i request, several numbers $\frac{\delta A_j}{\delta Req_{i_j}}$ in Formula 6 are greater than or equal to zero to establish the concept of similarity between the existing resource and the requested resource. In the flooding resource discovery element, it is even possible to specify which rows and columns in the matrix expressed in formula 6 must be greater than or equal to zero to establish the concept of similarity between the available resource and the requested resource.

In traditional computing systems, the concept of request is a concept regardless of the time and status of the system. In traditional computing systems, normally at the time, the request is made, either the requesting process enters the suspended state or the part of the process that attempted the creation enters the suspended state. Transferring the state of the process (or part of the process) to the state of suspension causes the request in this type of computational system to be a fixed concept independent of another independent variable other than the response. In Distributed Exascale Systems, the occurrence of a dynamic and interactive event may affect any of the beneficiaries of the resource discovery activity, including the request. The impact of a dynamic and interactive event on the request can affect the productive space, features, or operators of the request concept. This allows the request in Distributed Exascale Systems to be defined according to Formula 7.

$$f(Request) = f\left(time, system_{state}, D\&I, process_{plane}, beneficiary_{state}\right) f(Request) \overbrace{::}^{Define} < \overbrace{< \overbrace{RAS\left(time, D\&I, process_{plane}\right)}^{RAS\ independent\ variable} >}^{space\ definer}, \overbrace{Time_{Limitation}, Location_{limitation}, system_{conditation}, process_{plane}}^{attrinutes}, < \overbrace{Answer, Similarity, Structure, Review}^{avilable\ function} > > \tag{7}$$

As seen in Formula 7, the concept of request in Distributed Exascale Systems is due to the occurrence of dynamic and interactive events from a fixed concept to a time-dependent concept, system status, dynamic and interactive events effects, the interaction and communication plane of the requesting process with other processes, as well as other elements of the global activity of which the requesting process is a member, has become.

4.2 Request Framework According to Flooding RD in Distributed

In Distributed Exascale Systems, changing the pattern of request definition based on Formula 7 makes it possible to describe the concept of request based on the framework shown in Fig. 2.

As can be seen in Fig. 2, the focus of the request is Distributed Exascale Systems, in which the resource

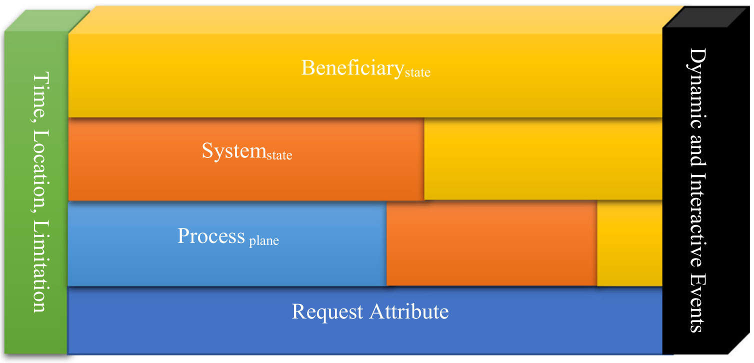

Fig. 2 Application framework leading to the call of the flooding resource discovery in Distributed Exascale Systems

discovery uses the flood pattern, which is based on the concept of the requested resource characteristics. If the distributed Exascale system manager uses a pattern similar to the operating system for resource management, then the requested resource properties are in the form $\ll Att_{memory_1}, \ldots, Att_{memory_n} >, < Att_{process_1}, \ldots, Att_{process_n} >, < Att_{file_1}, \ldots, Att_{file_n} >, < Att_{IO_1}, \ldots, Att_{IO_n} \gg$ is definable in which Att_{i_j} denotes j is the process request property attribute for the resource type i. In Distributed Exascale Systems, the characteristics of the request can be influenced by the constraints governing the request or the occurrence of a dynamic and interactive event. In Distributed Exascale Systems, the properties of the request may also affect the constraints governing the request or the occurrence of a dynamic and interactive event. Considering the effectiveness of the concept of request properties from the two constraints governing the request as well as dynamic and interactive events causes the requested feature to be described in Distributed Exascale Systems based on Formula 8.

$$\forall i, j \quad Att_{i_j} = \left(\sum_l D\&I_l \left| \left(\sum_k \left(Limitation_k | Att_{i_j} \right) \right) \right. \right)^{time}_{Request\ attribute} \tag{8}$$

As can be seen in Formula 8, for each attribute of the Att_{i_j} request defined by the process, it is possible to affect the dynamic and interactive event occurrence spaces as well as constraints. The request ruler obtains the request attribute Att_{i_j} According to Formula 8, for each Att_{i_j} request attribute, the effect of each constraint on the Att_{i_j} *an* attribute is first calculated. In Distributed Exascale Systems, such as traditional computing systems, each request can be a) Determining the time of response to the request b) Determining the acceptable time to search for the request c) Determining the place of responding to the request d) Considering limitations of resource discovery activity and e)Considering the nature of the request f) Considering the constraints governing the responder process g) Constraints governing the communication platform and the bottom layer h) Constraints governing the local management element of the requester and responder system.

In Formula 8, after calculating the internal sigma for the Att_{i_j} request, and extracting the request status after being affected by the above constraints and constraints, the effect of the constraints and constraints resulting from the occurrence of the event is investigated. It is dynamic and interactive on the concept of Att_{i_j} request. According to Formula 8, the traditional constraints and constraints governing the request, as well as the constraints and constraints resulting from the occurrence of a dynamic and interactive event, discreetly affect the concept of the application. The discrete effect of the two constraints and the mentioned constraints is due to the calculation of Formula 8 based on the two discrete concepts of time and also the nature of the request. From the point of view of the resource discovery, the mentioned constraints and restrictions change either according to the concept of the passage of time or according to the concept of changing the nature of the request. In traditional computing systems, the internal Sigma of Formula 8, based on the independent variable of time, is conventionally used to examine the effect of the constraints and constraints mentioned on the request. In distributed Exascale computing systems, in addition to the mentioned part, due to the possibility of changing the requested property, the external sigma part and defining the function mentioned in formula 8 based on the two variables of time passage and changing the requested feature, occurrence effects Giving a dynamic and interactive event to the request is calculated.

In Distributed Exascale Systems, in addition to the traditional constraints mentioned on the request, a dynamic and interactive event can cause the request to precede the request. Dynamically and interactively follow the status cycle shown in Fig. 3.

As shown in Fig. 3, the occurrence of a dynamic and interactive event may cause a) part of the request to change b) the request to become a new request c) the request to be unresponsive by the element resource discovery changes status d) The request is answered as a result of a dynamic and interactive event e) The request is not changed. In the first case, considering that from the point of view of the resource discovery element, the request properties are in the form $\ll Att_{memory_1}, \ldots, Att_{memory_n} >, < Att_{process_1}, \ldots, Att_{process_n} >, < Att_{file_1}, \ldots, Att_{file_n} >, < Att_{IO_1}, \ldots, Att_{IO_n} >>$ are described, so the occurrence of the event Dynamic and interactive data causes a limited number of Att_{i_j} to change. This change can only be done for one type of resource or several resource types. In this case, not all Att_{i_j} related to the request attribute description change. In the latter case, the occurrence of a dynamic and interactive event causes all Att_{i_j}'s to describe the request to be changed. In this case, the previous request becomes a request *Total Request*.In the third case, the occurrence of a dynamic and interactive event causes the constraints governing the request, both traditional constraints and constraints resulting from the occurrence of a dynamic and interactive event, to be changed in such a way that the resource discovery can do not respond to requests. In this situation, there is a condition (or conditions) that contradicts each other between the constraints governing the traditional request and the constraints arising from the occurrence of a dynamic and interactive event. These conditions (or conditions) make it impossible to calculate Formula 8.

In the fourth case, the occurrence of a dynamic and interactive event causes a condition (or conditions) to be applied to the request that satisfies the traditional constraints and restrictions governing the request, and in this case, there is no need to call resource discovery element. Any of the situations in which an existing request is converted into a partial or total modified request can be answered by the resource management element or lead to a situation where the resource management element is unable to calculate Formula 8. Defining the status of the request after the occurrence of a dynamic and interactive event relative to the previous request makes it possible to state that the conditions arising from the traditional constraints and restrictions governing the request with the constraints and constraints Due to the occurrence of dynamic and interactive events, they may not be in line with each other and cause each other to violate or be in line with each other. Resource discovery after a dynamic and interactive event occurs, can start resource discovery activities when it can calculate Formula 8 based on two independent variables over time or change the nature of the request. In the fifth case, the occurrence of a dynamic and interactive event is such that it either does not change the properties of the previous request or the result of the changes is such that the created request is no different from the previous request in terms of request properties.

As can be seen in Fig. 2, in Distributed Exascale Systems, unlike traditional computing systems, the central element of the system is the concept of global activity [22, 31]. Each global activity represents a set of processes that interact with each other. The interaction and connection between the processes that make up a global activity ultimately lead to the implementation of

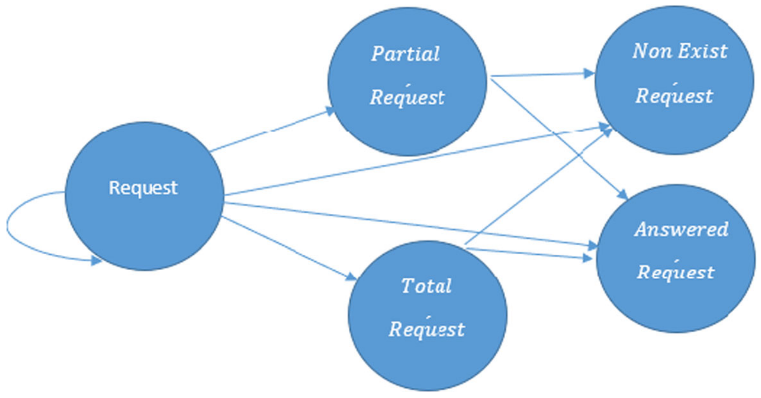

Fig. 3 Request situations after a dynamic and interactive event

an activity (or sub-activity) related to the scientific and applied program. From the point of view of the system management element, each set of computational elements in which the processes related to the global activity are executed is a structure in response to the scientific and applied program. An affine plane can be considered for any global activity and consequently any response structure. Each process in global activity is a member of an affine plane, in which the affine plane interacts with a set of processes and resources. The affine plane of which the process is a member is the Process$_{plane}$. In the context of a process request, the process plane indicates the requests that the process can respond to, as well as the requests that the process has compared to other processes. The processing plane indicates the resources that the process owns as of the process. The processing plane also contains resources that can meet the process requirements of the plane owner. If the Process$_{plane}$ owner's process can create a new Process$_{plane}$ process plane for each of its needs that intersects with the existing process plane and contains the process request, then the process request (s) will be answered. Conventionally in Distributed Exascale Systems, it is the responsibility of the resource discovery or the distribution management element to create a processing plane that can meet the needs of the process.

In Distributed Exascale Systems, since the structures respond to the standard requests of the process, which follow a pattern other than the dynamic and interactive event pattern, it is determined at the time of system design. Therefore, process planes responding to the normal request of processes are created at the time of system design. In Distributed Exascale Systems, process planes that respond to dynamic and interactive events are created during system execution by the resource discovery element. Process planes in Distributed Exascale Systems that are created based on the function of the resource management element are defined in the form (ReqAtt, ResVect, AnswerF). In the affine plane form created by the resource discovery in Distributed Exascale Systems, ReqAtt is a set of scalar sets containing the descriptive attributes of the requested resource, which are defined in the Request Attribute unit. There are four definable points per request, each point on a plane, so from the point of view of the resource management element, the ReqAtt scalar set creates four planes. The ResVect variable represents the set of process resource property vector vectors of the process plane owner and is a vector variable. Each vector member of the ResVect variable, if the vector is positive, then indicates the ability of the process to respond to other requests, and if the vector is negative, indicates the need of the process to use the resources of other processes. The AnswerF function is a resource discovery pattern tailored to the requirements of the process. In Distributed Exascale Systems, given that the pattern governing process requirements for managing dynamic and interactive events is different from other processes, it is possible to have a pattern for each process plane. Define unique in terms of resource discovery.

In the form (ReqAtt, ResVect, AnswerF) for the process plane, each process plane is based on four planes describing the requests of process, two sets of which include a) four response vector vectors and b) four-processor demand vector vectors It is a plane layout and is defined as a resource discovery pattern function to respond to dynamic and interactive event requests. For each requesting process plane, the two vector sets corresponding to the requesting process plane and the resource discovery pattern are created based on the process plane descriptor function described in Formula 9.

$$f\left(Process_{plane_i}\right) = \left[\left[ReqAtt_{1,1} \quad \dots \quad ReqAtt_{1,i} \right] \overset{Mapping}{\overbrace{\,|\,}} ResVect_i \right] \cdot \left[\begin{bmatrix} ReqAtt_{1,1} \\ \vdots \\ ReqAtt_{1,i} \end{bmatrix} \overset{Mapping}{\overbrace{\,|\,}} AnswerF \right] \tag{9}$$

As can be seen in Formula 9, the processing plate for the specified resource type i is multiplied by a vector space in a scalar space. Figure 4 shows the function of Formula 9.

As shown in Fig. 4, the vector space $[[ReqAtt_{1,1} \quad \dots \quad ReqAtt_{1,i}] \mid ResVect_i]$ Mapping the process request properties for resource i based on the corresponding vector of the process requirements of the process plane generator and vector space $\left[\begin{bmatrix} ReqAtt_{1,1} \\ \vdots \\ ReqAtt_{1,i} \end{bmatrix} \middle| AnswerF\right]$ Describes the process request properties mapping for resource i, based on the request-response pattern. From the point of view of the resource discovery element, the request properties space is converted to ReqF$_i$ scalar space and ReqAtt$_{1,j}$ vector space based on the two mentioned spaces, and the product of the interior of these two spaces creates a processing plane to process the request for resource i.

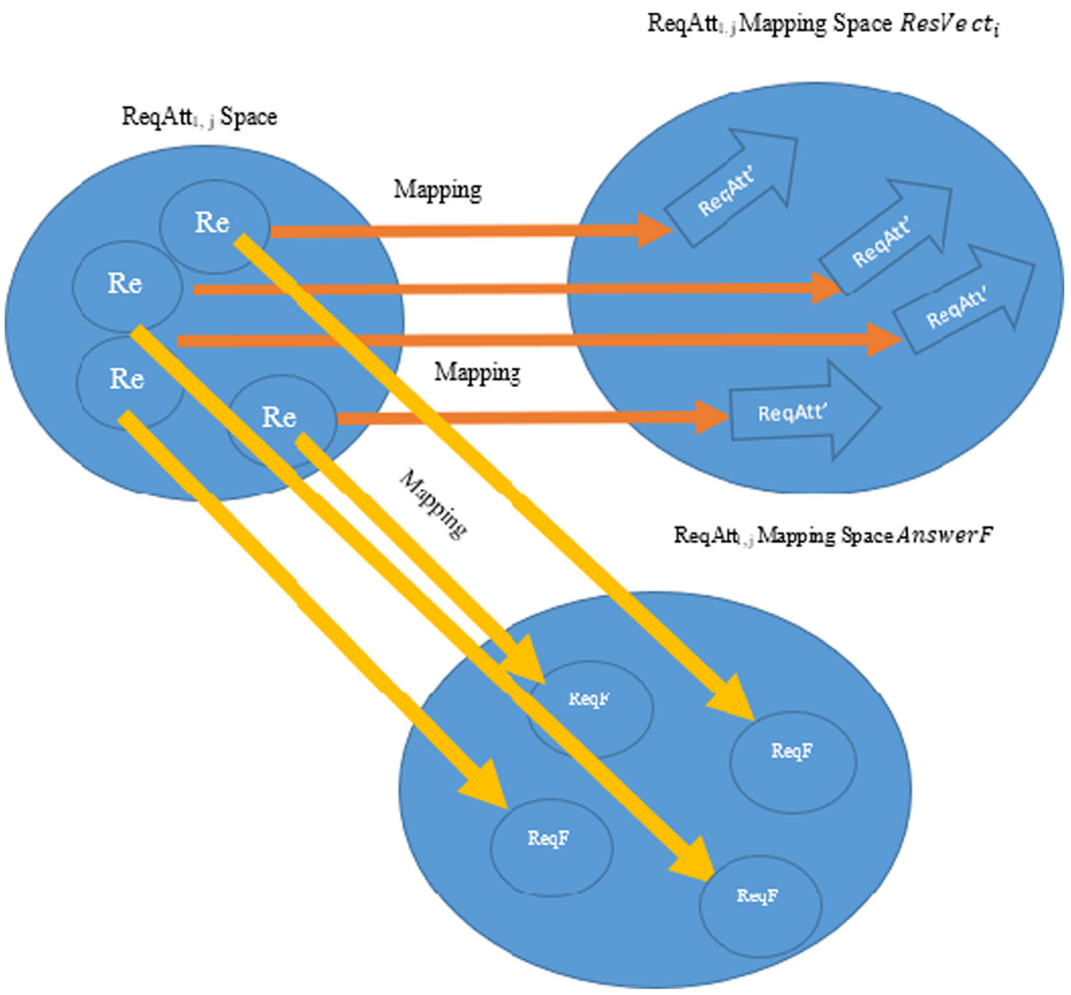

Fig. 4 Function of Formula 9 on creating a processing plane

N. In Formula 9, when $[ReqAtt_{1,1} \quad \dots \quad ReqAtt_{1,i}\ ]$ are mapped by the vectors corresponding to the process requirements to the vector space to be able to respond to a request ($ResVect_i$), in fact, each element $ReqAtt_{1,j}$ ($i \geq j \geq 1$) From the vector space to the described space, the property is mapped in terms of the vector. The result of this mapping is the definition of the vector $ReqAtt_{1,j}$, which describes the property of j's request for resource i based on the process requirements vector. In formula 9, when $[ReqAtt_{1,1} \quad \dots \quad ReqAtt_{1,i}\ ]$ is mapped by the AnswerF function, each element $ReqAtt_{1,\ j}$ from the scalar space to the space described by the attribute, after being answered by the function used by the resource management element, which is a scalar space, The result of this mapping is to define a new scalar value for each ReqF$_i$ element after being answered by the AnswerF function.

In traditional computing systems, on the process plane to request the process for resource i, the process vector requirements of the processor creating the plane and the request properties do not change after the plane is created. This makes it possible in this type of computational system to mention the vector space as well as the space of request properties as a constant and independent of the independent time variable. In traditional computing systems, the responsiveness vectors of requests change according to the ability of the computing element to respond to requests during the process of performing resource discovery activities and are considered as a variable of time-independent variables. Is taken. In traditional computing systems, according to the definition of plane generation in Formula 9, the process plane for requesting a process for resource i is not a function of time and is considered a fixed source during the execution of activities related to discovery. Based on what is stated in Formula 9, for each resource, a process plane can be defined for resource i response to request requests, based on the resource retrieval vector of requests. This plane is a time-independent variable based on both traditional computing systems and distributed large-scale computing systems.

In Distributed Exascale Systems, the processing plane for the resource requesting process i changes based on an independent time variable. A dynamic and interactive event can change any of the two sets of process request vectors or request properties. This causes that in Distributed Exascale Systems, both the processing plane and the resource plane of the computational element change based on the time-independent variable. In Distributed Exascale Systems, having four planes about the request and four planes about the resources in the computational element that can respond to the request allows the resource management element to be able to Decide on the status of the effects of a dynamic and interactive event on the request as well as the resources available in the computational element.

As shown in Fig. 2, in a distributed Exascale system, the state of the system affects the request and the characteristics of the request. The flooding resource discovery needs to consider the state of the system and its impact on the request related to the activation process of the flooding resource discovery element. In a distributed Exascale system, at any point in the process of creating a request by the process, the activator of the management element discovers the possibility of a dynamic and interactive event occurring and its effect on the system state and consequently the process request. There is. The flooding resource discovery element, based on the description of the system status, should be able to extract the effects made in the computing system space on the request of the process activating the flooding resource discovery element.

In Distributed Exascale Systems, the system manager typically describes the state of the system based on four spaces (SAtrib, MAtrib, AAtrib, Time) [32]. In the template used to describe the state of the system, SAtrib represents the properties used to describe the state of the system. In the flooding resource discovery mechanism, the process of requesting the activation process of the resource discovery is based on the RAS concept, so the flooding resource discovery must be able to redefine the concept of system properties based on the RAS concept to make changes. Due to the influence of the computational system on a dynamic and interactive event on the characteristics of the decision request. From the point of view of the flooding resource discovery element, the definition of the system is different from the definition of the computational system in which the request is formed. From the point of view of the flooding resource discovery element, the system is equal to the set of computational elements that affect the request. Any computational element that is a member of the global activity of which the applicant process is a member or any computational element that is not a member of that global activity but in which a process (or processes) affect the performance of the applicant process, they are the elements of resource discovery. According to this definition, the element (or elements) that influence the concept of the applicant process in the resource discovery process is also a member of the system.

The MAtrib variable indicates the properties of the computational element in which the request is located. Description of the features of the computing system, from the point of view of the flooding resource discovery element, should be redefined based on the characteristics of the request to describe the status of the request. The MAtrib variable is an event-dependent variable from the point of view of the flooding resource discovery element. This variable is not the only one that describes a computational element. In the flooding resource discovery mechanism, the flooding resource discovery is responsible for requesting the activation process for the resource discovery. This causes the request to be transferred to that computational element each time it is checked outside the system. The event that changes the computational element containing the request is the non-response of the computational element to the request and the transfer of the request from one computational element to another computational element based on the flooding resource discovery mechanism.

The variable AAtrib indicates the characteristics related to the main global activity of the system. From the point of view of the management element, the discovery of the resource of global activity is defined based on space (AAtribReq, AAtribAns). The AAtribReq space represents the global activity properties that the process activates as a member resource discovery element. The AAtribAns space represents the global activity characteristics created by the flooding resource discovery to respond to the process request. During the process of implementing activities related to each of the two global activities, these two spaces may have an impact on each other. From the point of view of flood management, the impact and effectiveness of these two global activities on the centrality of the RAS concept are examined.

The flooding resource discovery uses formula 10 to define the state of the system relative to the request.

$$\textit{System State Define based on Request} = \left[\left(\frac{\delta RAS(E,t)}{\delta SAtrib(E,t)}\right), \left(\frac{\delta RAS(E,t)}{\delta MAtrib(E,t)}\right), \left(\frac{\delta RAS(E,t)}{\left(\frac{\delta RAS(E,t)}{\text{AAtribReq}(E,t)}\right), \left(\frac{\delta RAS(E,t)}{\text{AAtribAns}(E,t)}\right)}\right), \left(\frac{\delta RAS(E,t)}{\delta t}\right) \right] \tag{10}$$

As can be seen in Formula 10, the flooding resource discovery redefines the system state descriptor space based on the partial derivative of each of the system state descriptor spaces relative to the RAS variable. The concept of RAS is considered as describing the concept of the resource requested by the process based on the two variables of event and time. Time refers to the passage of time governing the element of flooding resource discovery. The event can include a change in the execution location of the resource discovery or any change that alters the function of the flood resource management element. The partial derivative of RAS concerning each of the four spaces describing the state of the system causes the said space to be rewritten according to the concept of RAS and to express the changes of the mentioned space according to RAS. Rewriting each of the constituent spaces of the RAS-based system state description allows system state changes due to dynamic and interactive event occurrence in terms of RAS to be examined. In this case, when a dynamic and interactive event occurs in the system because the independent and descriptive variable of the system status is equal to the resource property of the requested process, so describe the impact of the dynamic and interactive event in terms of the independent variable of the resource attribute requested by the process.

If the partial derivative of *RAS(E,t)/X*, where X can be a member of the spaces (SAtrib, MAtrib, AAtrib, Time), is a value greater than zero, then it is possible to define the system state based on RAS. If the value of the partial derivative mentioned is less than zero, then it is not possible to define the system status based on RAS. If the value of the partial derivative mentioned is equal to zero, then the RAS value is defined for a specific moment of checking the system status and there is no requirement to define the system status based on RAS for other times of the system execution process. In the case where the partial derivative equals zero, the event is derived from formula 10 based on the event independent variable, and the time-independent variable is considered as a constant. This allows the RAS system definition to be described based on events that change the system state. In Formula 10, each of the four system state description spaces can have different partial derivatives of each other, or they can all follow the same partial derivative pattern. Because the four spaces that describe the state of the system are separate from each other, the pattern governing their derivative can also be separate from each other. Defining partial derivatives for each of the spaces describing the state of the system means that the space can be described according to the RAS concept. The flooding resource discovery can describe the state of the system in the architecture shown in Fig. 2 if it can describe all four spaces describing the state of the system based on the RAS concept.

As shown in Fig. 2, in Distributed Exascale Systems, defining the applicant process based on the concept of global activity results in a set of beneficiaries for each request. Applicant process beneficiaries are influential in defining the RAS of the application. The set of interactions and connections between the requesting process and the beneficiary processes of the requesting process changes the RAS space. For each interaction and connection between the activator process and other stakeholder processes, an affine plane can be considered as described in [32]. For each affine plane, a Latin square corresponding to the relationship and interaction of the two processes can be defined. It can be proved that for each Latin square describing the interaction and relationship between two processes, given that one of the two processes necessarily activates the flooding resource management element, a vector such as $C_{l,zi}$ can be defined where l represents the process properties that activate the flooding resource discovery that interacts with other processes. The zi index of process properties interacts with the process. For per $C_{l,zi}$ matrix, a sequence of k non-zero vectors c_1 to c_k can be defined whose values are equal to the rows of the $C_{l,zi}$ matrix. The flooding resource discovery uses Formula 11 to decide on the outcome of the linear combination of interactions between two processes and, consequently, the outcome of the linear combination between processes with the activating process of the resource management element.

$$\text{Communication}_{\text{space}} = \left[\sum_{p=1}^{l} \frac{\left[\sum_{k=1}^{zi} \left[\frac{\left(Communication_{space} | c_k \right)}{\|c_k\|^2} \right] \right]}{[\![l_n]\!]^2} c_k \right] (t)$$
$$(|\ \text{means to divide,}\ \ \|A\|\ mens\ norm\ of\ \ vector\ A) \tag{11}$$

As can be seen in Formula 11, the communication and interaction space of the process beneficiaries is obtained by calculating the linear composition vector. In Formula 11, the internal matrix calculates a linear combination between the process vector requesting the resource management element with each of the other processes. In internal sigma, the presence of the definition condition ($Communication_{space}|\ c_k$) reduces the space of the $C_{l,zi}$ matrix. The two processes that activate the resource discovery element, as well as the other process, may be unrelated to some of the properties of each, so the value of the corresponding matrix element in them is zero, there is a definition condition ($Communication_{space}|\ c_k$) causes the mentioned data not to be considered. On the other hand, the specified non-zero vector sequences $c_k$in the internal sigma represent the interaction vector and the relationship between the process that activates the flooding resource discovery with other processes. The vector obtained from c_k is a vector in the positive direction, the size of which indicates the importance of the interaction and relationship between the process that activates the resource management element with a specific process.

The c_k vectors related to the resource activation process with other processes can create an interactive process affine plane. In the interactive end of the process, if the points represent the processes and the line represents the interaction and communication, then a) exactly one line passes through both points. B) Both lines intersect only in the process that activates the flooding resource discovery element. Internal sigma and endgame creation are obtained for each other process that interacts with this process. Considering the dynamics and interaction of inter-process communications and interactions, it can be concluded that at any given moment a set of the affine plane is created for the process of activating the flooding resource discovery. Participates in them as a point. In the interactive end of the process, the lines are oriented because they are derived from the c_k vectors and their direction indicates the direction of the interactions. In Formula 11, in the inner sigma, the value $\|c_k\|^2$ is the size and significance of the interaction and communication vector for the process that activates the resource management element with other processes.

In Formula 11, the outer sigma represents the beneficiaries plane governing the process that activates the resource management element. The external sigma is obtained based on the columns of the $C_{l,zi}$ matrix. In all intermediate matrices that activate the management element, the discovery of a flooding resource with any other process that interacts with it is assumed that the matrix columns represent the properties of the process that activate the management element. Discovery is the source of floods that are effective concerning other processes and are present in all connections between this process and other processes. This makes it possible to obtain the result of all the affine planes in which the activation process of the resource discovery is based, based on the external sigma.

This affine plane shows all the interactions and connections between the mentioned process with any other process that is somehow related to the said process. In affine plane obtained based on the external sigma, the interactive and communication space may include more than If there are two processes, in this case, the other two processes interact with each other, which is defined based on the process activating the flooding resource discovery element. In this situation, the mentioned interaction is of the type of interactions and communications related to the global activity. In interactions and communications between global activities, if four processes are considered, then the line of interaction and communication does not include four points and certainly includes three points. This principle implies that because the interactive affine plane is intended for the process that activates the flooding resource discovery element, any interaction, even if it has more than two stakeholders, must be in the form of (other processes, active process). Flooding resource discovery (other processes) to show the interactions and connections between the process activating the resource discovery with other processes. In Formula 11, the value $[\![l_n]\!]^2$ indicates the significance coefficient of the concept of interprocess interactions for the process that activates the resource management element. Formula 11 is based on the independent time variable. The reason for this is

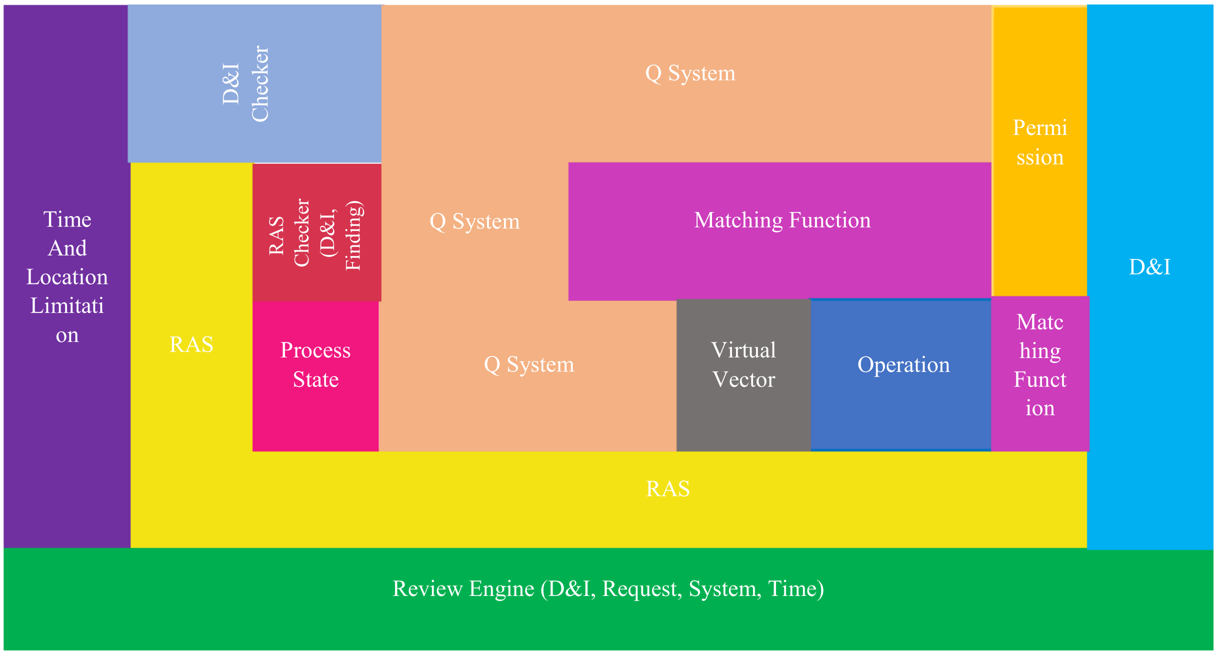

Fig. 5 ExaFlooding RD architecture

the nature of the variability of interactions and connections between processes, whether at the level of one process with the process activating the resource management element with another process or at the level of interactions and communication between other processes and the process activating the resource management element. The flood returns.

In Formula 11, it is stated that the beneficiary of the management element activating process must have an interaction with the source discovery activating process. In this formula, the assumption is based on the fact that if a process that interacts with the process activating the management element is flooding resource discovery, then it can be considered as an influential or influential process on the process activating the management element. Resource discovery to be considered.

4.3 ExaFlooding RD Framework

In Distributed Exascale Systems, at any point in the process of implementing activities related to the element of flooding resource management, the possibility of dynamic and interactive events occur in any of the relevant areas. There is activity to the flooding resource discovery. In traditional computing systems, the scope of activity of the flooding resource discovery includes the computational element containing the process activating the flooding resource discovery element, and the computational element capable of responding to the process request. In activities related to the function of the flooding resource discovery element, the focus of the activity is based on the concept of request. As shown in Fig. 1, in Distributed Exascale Systems, there is no requirement to respond to a request by a single computational element. In this type of computing system, the function of the ExaFlooding RD, and the concept of Matching Function, may create a global activity to respond to the request of the ExaFlooding RD activating process. This causes the system governing the ExaFlooding RD, unlike traditional computing systems, which consists of only two requesting and responding elements, to include a set of computational elements, each of which is a part (or part). Are responsive to the request for payment.

Developing the concept of ExaFlooding RD activity, defining the concept of global activity and responding to part (s) of the ExaFlooding RD activation process request, considering the concept of Matching and developing it based on the concept of Similarity, as well as the need to consider management A dynamic and interactive event at any point in the execution of activities related to the ExaFlooding RD allows an architecture similar to that shown in Fig. 5 to be considered for the operation of the ExaFlooding RD.

As can be seen in Fig. 5, the focus of the functional architecture of the ExaFlooding RD is based on the concept of reviewing the resource discovery activity. The function of the ExaFlooding RD is such that a dynamic and interactive event can occur at any point in the process of performing resource discovery activities and even after the resource is allocated to the requesting process. The occurrence of a dynamic and interactive event may alter the status of any element influencing or affecting the performance of the ExaFlooding RD in such a way that either the execution of the activity fails or the execution of the activity is meaningless or the variables need to be revised. The exact functionality of the ExaFlooding RD to perform the activity. Considering the Q System as the functional system of the ExaFlooding RD, it can be stated that the occurrence of dynamic and interactive events, changes made in the request as well as changes made in the system of global activity can be the activating factors that need to be considered in the process of implementing activities related to the ExaFlooding RD.

In traditional computing systems, the flood resource discovery with the focus on the request process, after finding the computing element that can respond to the request, receives the resource allocation activity from the local management element and then assigns a resource responsive to the request, completes the resource discovery activity. In traditional computing systems, since the elements of the flooding resource discovery system do not change the status of the elements, there is no need to review and control the activities related to the requesting process, after allocating the resource. Failure to change the computational element containing the requesting process and consequently the requesting process, as well as not changing the status of the computational element responding to the request and consequently not changing the resource allocation pattern to the process, causes the flooding resource discovery to manage post-event situations to be assigned. The presence of the D&I element in Fig. 5 provides this capability for the ExaFlooding RD to be able to handle the dynamic and interactive event described in [1] decide. If based on the patterns defined in the D&I Checker element, this element detects a dynamic and interactive event occurring, this will cause the Review Engine element to be called by the D&I Checker element. And review the implementation process of the activities of the ExaFlooding RD. The D&I element starts its activity based on the three main patterns of dynamic and interactive events described in [1]. During the implementation of activities related to the ExaFlooding RD, the D&I element, as a result of dynamic and interactive event occurrence, extracts the pattern related to dynamic and interactive event occurrence. Given the possibility of a dynamic and interactive event occurring in a distributed large-scale system and its impact on any of the activities (or activities) defined in the system, the D&I element examines patterns or occurrences. Data that affect the performance of the ExaFlooding RD. The D&I element, from the set of dynamic and interactive events that occur in the system, only examines the pattern or event leading to the dynamic and interactive event that affects the function of the ExaFlooding RD and consequently the Q System.

Detecting an event or a dynamic and interactive pattern in the D&I Checker element may cause the Resource Attribute Set to change. In this case, the occurrence of an event or dynamic and interactive pattern has affected the activation process of the ExaFlooding RD and has caused either the set of features related to the request of the ExaFlooding RD activator to be changed or the set of features related to ExaFlooding RD activator request to be canceled. The occurrence of either of these two issues causes the ExaFlooding RD to need to review the request attribute set and execute activities related to the ExaFlooding RD functionality function based on the new request attribute set.

Occurrence detection and dynamic and interactive pattern in the D&I Checker element may cause either by changing the settings of request descriptor features or by reviewing the activities of the ExaFlooding RD and considering its new functional function and consequently Considering the restrictions and restrictions expressed for the request in the Time and Location Limitation element, the ExaFlooding RD cannot find the requested source with 100% properties that match the properties expressed in the requested resource feature space element. In such a situation, the D&I element calls the Matching element to use the Matching Function to find the source whose properties are most similar to the properties described in the space describing the properties of the requested source.

The definition of the concept of Matching Function and the concept of process request of the ExaFlooding RD makes it possible to respond to part (or parts) of the request of the process in the computational element outside the system. Considering the concept of Matching Function makes the model whether the computing element can join the Q System? Compared to traditional computing systems. In traditional computing systems, if the condition for normalization of the minimal RAS polynomials by Machine polynomials is met, W represents the function of the flooding resource discovery management function. This is even though in the ExaFlooding RD, part (or parts) of RAS polynomials can be normalized by minimal Machine polynomials. The normalization of the part (or parts) of the RAS polynomial can cause a part of the linear operator W to represent part of the function of the ExaFlooding RD in the computational element. In such a situation, the Q vector, in addition to being able to contain computational elements that are fully capable of responding to RAS, can also be computational elements that are responsible for different part (or parts) of RAS. Responsiveness to different RAS part (s) requires redefining RAS polynomials from the scalar form $f(RAS) = \alpha_1(IO)^{n_1} + \alpha_2 (File)^{n_2} + \alpha_3(Memory)^{n_3} + \alpha_4(Process)^{n_4}$ to form $f(RAS) = \overrightarrow{\sum\alpha_1(IO)}^{n_1} + \overrightarrow{\sum\alpha_2(File)^{n_2}} + \overrightarrow{\sum\alpha_3 (Memory)^{n_3}} + \overrightarrow{\sum\alpha_4(Process)^{n_4}}$. Any vector $\alpha_i(X)^{n_i}$ is a vector in the positive direction whose size is equal to the amount of resource required by X. In any computational element where a part (or parts) of a RAS polynomial is accrued by Machine polynomials, the vector is the size of the resource that can be allocated to the part (or parts) of the request by the computational element responding to the request. It is created by the coefficient of the importance of the response of the respondent to the request and the number of resources required to respond to the respondent. The result of the vectors for the specified source X in the process of implementing the global activity must be equal to the total amount of source required for the request for source X and its positive direction, and the sum of the scalar coefficients of importance must be equal to the coefficient required for the application. Similarly, machine scalar polynomials must be formed $f(Machine) = \overrightarrow{\sum\beta_1(IO)^{m_1}} + \overrightarrow{\sum \beta_2(File)^{m_2}} + \overrightarrow{\sum\beta_3(Memory)^{m_3}} + \overrightarrow{\sum\beta_4(Process)^{m_4}}$ Become. The direction of the vector $\beta_i(X)^{m_i}$ is in the negative direction and its size is equal to the source that can be allocated to the process in the computational element responding to the request.

Redefining the RAS and Machine vectors based on the concept of vectors provides the capability for the ExaFlooding RD to occur when a dynamic and interactive event occurs in any of the elements that make up the framework shown in Fig. 5 and change. The direction of the vector, whether changing the direction of the vector in the opposite direction of the response process or creating an angle between the two vectors RAS and Machine, which means reducing the response capability in the coefficient of importance or number of resources, decide.

Considering the redefinition of RAS and Machine polynomials and the possibility of considering the normalization of a part of RAS polynomials by minimal Machine polynomials, the possibility of defining the function of the ExaFlooding RD, based on the concept of normalizing part of a polynomial With the RAS vector exists by polynomials equivalent to the Machine vector. According to the concept of addition of Machine and RAS vectors in each computational element, the condition that the computational element can be a member of the Q System is that in addition to normalizing part (or parts) of polynomials equivalent to the RAS vector, the result of the vector difference $\overrightarrow{\mathrm{RAS}} - \overrightarrow{\mathrm{Machine}}$.

It must also have a non-zero value. Vector difference value indicates the partial function of $W_{i,j}$, where i represents the part (s) answered by the computational element to the process request, and j represents the computational element. As a result, in the vector difference $W_{i,j}$, the value of i can contain one or more of one type of source, indicating that in the case of several types of sources, part (or parts) of the processing request by the computational element j Answered

$$\overrightarrow{W} = \sum_{j=1}^{n}\left[\sum_{i=1}^{m}\overrightarrow{A_{jFile_i}}q_i,\quad \sum_{i=1}^{m}\overrightarrow{A_{jMemory_i}}q_i,\quad \sum_{i=1}^{m}\overrightarrow{A_{jProces_i}}q_i,\quad \sum_{i=1}^{m}\overrightarrow{A_{jIO_i}}q_i\right] \tag{12}$$

As can be seen in Formula 12, this extended formula is Formula 2, considering the ability to respond to part (or parts) of the request by the computing elements of the Q System member. In Formula 12, in internal sigma, vector $\overrightarrow{A_{jX_i}}$ which X can be one of the IO, Memory, File, or Process values, denoting the vector (or vectors) that reduce the RAS vector in the j element. The vector $\overrightarrow{A_{jX_i}}$ implicitly includes the ability to respond to X-type requests, which could be the ability for one or more parts of an X-type request. Also vector$\overrightarrow{A_{jX_i}}$ Including the ability to access resource rights, resource allocation time, concurrent use of the resource by local and global processes, resource reusability, as well as the creation of structured resource-based response structures and dynamic and interactive events that The resource owner's process can manage and can continue the process of responding to the request vector $\overrightarrow{A_{jX_i}}$ In the function of ExaFlooding RD, in addition to considering the ability to respond to the request, the ability to create a response structure can maintain information about dynamic and interactive event management that affects the performance of the resource and especially the request-response process by the resource Are, is. Information about what patterns or events occur by the resource owner process shown in the D&I element in Fig. 5 ensures that the source response process is not challenged. Having dynamic and interactive event information enables the source owner process to call the D&I Checker as well as the RAS Checker to manage the execution of the W function related to the ExaFlooding RD function in such a way that the process request is answered.

RAS Checker unit reviews RAS based on the dynamic and interactive event and considers new spatial and temporal constraints. In the function of the ExaFlooding RD, dynamic and interactive event occurrence may cause constraints to occur. And change the restrictions on the properties of the requested resource. This change in the constraints on the requested resource properties can be due to the effect of a dynamic and interactive event on the process of activating the resource management element or due to the influence of the ExaFlooding RD in each activity. Be related to the source discovery process.

In traditional flood-based resource discovery mechanisms, the three elements of resource discovery, resource access rights, and resource allocation to the requesting process can be considered for the resource management element. For the ExaFlooding RD, in addition to the activities intended for it, due to the possibility of defining the concept of global activity as well as dynamic and interactive event occurrence, activities such as removing the computational element from the Q System can be redefined. The response of the computational element to a part of the request was to redefine the global activity based on the capabilities created in the computational elements after the dynamic and interactive event occurred, as well as to review the need to continue implementing resource discovery activities. The Operation Unit contains the current activity of the ExaFlooding RD based on system conditions and the process of performing resource discovery activities.

In the ExaFlooding RD, the Q vector is converted to the algebraic sum of the Q vectors for each computational element that can respond to the part (or parts) of the request. Due to the dynamic and interactive event occurring, in the ExaFlooding RD, the Q vector is considered as a function of time. In this vector, the computational element may have two conditions for membership in the Q System, but due to a dynamic and interactive event, the status of the computational element changes in such a way that it is not possible to continue in the response process. This causes the two conditions of membership of the computational element in the Q System to be a function of the dynamic and interactive event occurrence defined in the D&I element. In such a situation, the computational element at the moment t = omega and when receiving the request of the ExaFlooding RD was able to meet the two conditions for responding to part (s) of the request, but at the moment t = omega + ε occurs Giving the dynamic and interactive event defined in the D&I element shown in Fig. 5 has caused either the state of the computational element to change in such a way that it can satisfy two conditions and normalize part (or parts) of The polynomial is not equivalent to the RAS vector, or the request state has changed in such a way that the computational element can't establish two conditions. If the inability to respond to the request is due to the occurrence of a dynamic and interactive event in the computational element, then the Review Engine unit is called and, consequently, the Matching unit and the D&I Checker unit are called. Takes.

A review of the Matching unit's ability to respond to a part (or parts) of a request, as well as the management of a dynamic and interactive event and its impact on the computational element, may allow a part (or parts) of a response to be requested again. Otherwise, the Q System unit will remove the computational element from the set of elements that make up the Q System. This change in Q System status necessitates a change in the status of the Operation element. If the inability to respond is due to a dynamic and interactive event affecting the request, this change will call the Review Engine and consequently the RAS Checker, if the RAS has changed and the change is somehow It is not possible to respond to the part (or parts) of the request based on the Matching element, in this case, while changing the RAS and removing the computational element from the Q System, the status of the effective and affected elements of the Q System will change according to the new RAS. Slowly, and by considering the time and space constraints as well as the D&I patterns, a new state of the search system is created.

In the ExaFlooding RD, for each computational element that responds to the request, the vector $QM_{i,j}$ can be defined, which indicates the ability of the computational element of the Q System member to respond to the process request that has the status of Process State.

The vector $QM_{i,j}$ is part of the global activity vector described in the Q System. Like traditional computational systems, ordered base such as $\{q_1, \ldots, q_n\}$ can be defined for the vector Q. On the regular basis, q_i represents the Matching coefficient and part of the Matching Function in Fig. 5. For each element that makes up the global process request-response activity, one or more q_i are generated, the algebraic sum of which together represents the q_i of the computational element of the global process request-response activity member. Considering that n computational elements respond to the process request, the regular basis $\{q_1, \ldots, q_n\}$ can be defined for the vector Q. In Q System, the value of q_i is a number between zero and one, which is never equal to zero, and represents the similarity coefficient to the source described in RAS, in the computational element i, and is calculated based on the Matching Function. The Matching Function element creates a $QM_{i,j}$ matrix for each computational element. The corresponding matrix, or $QM_{i,j}$, is in the form of a 4 * N matrix, each element of which represents the similarity coefficient between the specified capacity of the computational element and part of the processing request.

The $QM_{i,j}$ matrix is obtained by the Matching Function element on the basis that a row for the matrix is considered for each specific resource of file, memory, IO, and Process. The members of line i represent the capabilities of the computational element in terms of a specific source type, taking into account time and space constraints. The request received by the computational element is also described based on the RAS element, the space of the requested resource properties as well as the constraint element and time constraints. And time constraints are considered as points x_1 to x_n, and each x_i is part of the processing request whose information is described in the requested resource properties space. The information in the requested resource attribute space and the request description can also be considered as dots x_1 to x_n in a matrix containing 4 rows, each row of which refers to the definition of a part of the request for a specific resource type. This matrix is called the RM_{ij} matrix. The RM_{ij} matrix is generated for each request leading to the ExaFlooding RD being called, while the $QM_{i,j}$ matrix is created for each computational element examined by the ExaFlooding RD. The Matching Function element calculates the similarity between the two mentioned matrices according to formula 13.

$$\forall QM_{ij} \ and \ RM_{ij} \ \overbrace{::}^{define} \ Similarity\left(QM_{ij}, RM_{ij}\right)$$
$$= Similarity\left(RM_{ij}, QM_{ij}\right)$$
$$= Similarity \left\| QM_{ij} - RM_{ij} \right\| > P \ \ Eq.13$$

As can be seen in Formula 13, in the Matching Function element, the Matching function uses the similarity function [33] to find the $QM_{i,j}$ capability in the capability matrix of the computational element that is similar to the RM_{ij} part of the request is. Given that the range of the similarity function is positive and its value is a number between zero and one, the ExaFlooding RD can define a concept called the acceptable similarity limit or P, which in case of similarity between the $QM_{i,j}$ element of the computational element power matrix with the RM_{ij} section of the P value request, the computational element responds to the RM_{ij} request based on the $QM_{i,j}$ capability. The functional model of Formula 13 is such that for any given value $QM_{i,j}$ this value with all values of RM_{ik} whose i is the same and k related to RM_{ik} varies from one to j, in terms of similarity And if the specified value of $QM_{i,j}$ with one of the RM_{ik} has a similarity of one or more similarities than P, then based on the ExaFlooding RD, the computational element can respond to a part of the request with a specified value of $QM_{i,j}$.

In the function of the ExaFlooding RD, the D&I element decides on the occurrence of dynamic and interactive events affecting the function of the ExaFlooding RD by examining the ongoing events in the process of executing activities related to the ExaFlooding RD.

As shown in Fig. 5, the D&I Checker element overrides the Q System as it is a dynamic and interactive event that has affected either the Q System or the Process State. Or RAS by considering time and space constraints. In the operation of the ExaFlooding RD, the D&I element detects the occurrence of a dynamic and interactive event either based on the initial patterns of dynamic and interactive event occurrence or based on the patterns obtained in the process of executing the ExaFlooding RD.

The D&I element, due to its connection to the access rights element, can detect the occurrence of a dynamic event and the interaction that leads to a change in access rights to the resource. The occurrence of a dynamic and interactive event affecting the request changes the requirements of the request for access to the resource. On the other hand, the occurrence of a dynamic and interactive event affecting the resource can also lead to a change in the status of access rights to use the resource.

The D&I element is also associated with the Matching Function element. Calling the Matching Function element means that the Q System is not able to respond to the processing request, taking into account the constraints and restrictions specified in the Time and Location Limitation. From the D&I element's point of view, a dynamic and interactive event causes either the constraints and temporal and spatial constraints of the request to change or the characteristics of the requested source to change. The association of the D&I element with the RAS element results in an event pattern that occurs in the event of a dynamic and interactive event that alters the RAS associated with the request. The relationship between the D&I element and the set of elements enables the D&I element to extract dynamic and interactive event occurrence patterns, including patterns leading to a dynamic and interactive event affecting the respondent and source.

Extracting dynamic and interactive event patterns causes the D&I element to invoke the Review Engine element if a pattern or dynamic and interactive events occur. Calling the Review Engine element causes the D&I Checker unit to be activated due to the specificity of the pattern leading to a dynamic and interactive event. If the occurrence of a dynamic and interactive event has affected the request, then the Review Engine unit will call RAS Checker based on the dynamic and interactive event. If a dynamic and interactive event changes the status of the source, then the Review Engine unit will call D&I Checker to change the Q System and Matching Function to manage the dynamic and interactive event.

In the function of the ExaFlooding RD, the D&I element generates the α_i vector for each activity performed by the ExaFlooding RD, the size of which indicates the importance of the activity performed by the ExaFlooding RD in discovering the source and direction. It is in the positive direction of the unit vector. If it is assumed that at the moment t = alpha a dynamic and interactive event has occurred, then

A) Leads to a change in the status of the process activating the resource management element, and in particular the status of the request. In such a case, the request extends from <*RAS* > , *Time*$_{Limitation}$, < *Answer*, *Similar* > >to the form shown in Formula 7. Extending the concept of request from the form expressed to the form shown in Formula 7 makes the concept of request change from a concept independent of the place-independent variable to a time-and-place-dependent variable, system status, dynamic event occurrence, and become an interactive, interactive, and communication plane of the process with other processes as well as the beneficiaries governing the process in the global activity.

In Formula 7, contrary to the traditional definition of request, the concept of request is not considered as an abstract concept independent of other concepts. In traditional computing systems, the occurrence of a request in a process is traditionally considered as an abstract concept that responding to it will continue the process of processing the process. This is even though in Distributed Exascale Systems, the concept of processing is considered as part of the global activity. Considering the process as part of the global activity makes each process have a set of interactions and connections with the member processes of the global activity. The existence of a set of interactions and communications, as well as the impact of the process on other processes that are members of the global activity, causes that at the time of the request, the concept of the created request is also affected by interactions and communications, as well as the impact of the process owner.

In Distributed Exascale Systems, dynamic and interactive event occurrence may also change the status of the request, the constraints governing the request, and even the characteristics of the requested resource. Defining a request in a global activity space and the need to consider the concept of dynamic and interactive events makes the RAS concept move from an abstract concept to a time-dependent concept, a dynamic and interactive event as well as the status of the request owner process. In interactions and communications with other member processes, change the global activity. The dependence of the RAS variable on the listed variables necessitates the need to review the RAS variable in the architecture of the ExaFlooding framework and the need to consider the effects of the independent variables defining the RAS. Flooding resource discovery for each dynamic and interactive event, by changing the communication and interactive plane of the requesting process as well as the time of the flooding resource discovery management system, review the RAS and its properties. Requests activate the flood resource discovery element. In Distributed Exascale Systems, unlike traditional computational systems, where there is only a time limit for responding to a request, the constraints for responding to a request are considered quadruple $<Time_{Limitation}, Location_{limitation}, system_{state}, process_{plane}>$

Taking into account the status of the system means considering the impact and effectiveness of the elements in the computing system containing the requesting process on the request. In the four mentioned above, the process plane also includes the interactions and communications of the process with other processes that are members of the global activity and the impact of the global activity on the request and, consequently, the response to the request.

Developing the concept of request based on Formula 7 allows the functionality of the ExaFlooding RD to be developed in comparison with traditional computing systems. In Distributed Exascale Systems, the ExaFlooding RD, in addition to finding the computational element that can respond to the request, also creates a new response structure for managing dynamic and interactive events.

The creation of a response structure for a request is based on the concept of the inability to respond to a request by a specific computational element or due to the lack of a precise definition of RAS. Inability to respond to the request the activating element of the flooding resource discovery may be due to the complex nature and structure of the RAS or the existence of constraints on the response to the request. In Distributed Exascale Systems, due to the definition of a quadruple space for request-response constraints, situations may arise in which the ExaFlooding RD creates a response structure to respond to the request.

In distributed large-scale systems, failure to accurately define RAS or change RAS in terms of time-independent variables, dynamic and interactive events, as well as the interactive and communication plane of the requesting process with other processes may result in the flooding resource discovery lacks a precise definition of RAS. In Distributed Exascale Systems, on the other hand, changing the responsiveness constraints may also cause the ExaFlooding RD to find the source whose RAS exactly matches the RAS element of flooding resource discovery. A resource whose source RAS is similar to the RAS is required by the flooding resource discovery. This makes the Similarity activity based on the Matching function defined as one of the most conceivable activities for the ExaFlooding RD. In the Similarity activity, the ExaFlooding RD, for whatever reason, including changing the response constraints, cannot find the computational element containing the source whose RAS exactly matches the RAS required by the flooding resource discovery element. Is. This issue in case of a dynamic and interactive event and its effect on the description of the request status in distributed large-scale systems, especially request constraints, as well as in case of dynamic and interactive event occurrence and feature change the descriptors of the requested resource may also cause the ExaFlooding RD to use the concept of similarity for 100% RAS compliance.

In Distributed Exascale Systems, the definition of the concept of request based on the independent time variable, and especially the dynamic and interactive event, makes it possible to change at any point in the implementation of activities related to the ExaFlooding RD. The characteristics of the request, the constraints governing the response to the request, and even the activity running by the flooding resource discovery are present. This allows the ExaFlooding RD framework architecture, in distributed large-scale systems, to have an activity review element that works directly with RAS, request, dynamic event occurrence, and is interactive as well as temporal and spatial constraints. The revision element changes or does not change the operation element for any change in one of the four spaces <Time, Request, D&I, System>.

B) The occurrence of a dynamic and interactive event may result in a change in activity or a change in the status of the element (or elements) responding to the request after the global resource discovery activity. In the event of a dynamic and interactive event, the functional function of the flooding resource discovery may change. As can be seen in Formula 1, the centrality of the function definition function of the flooding resource discovery is based on the process requirements or RAS. In distributed large-scale systems, the occurrence of a dynamic and interactive event on the requesting element changes each of the two process requirements spaces and describes the characteristics of the requested resource. In distributed large-scale systems, given that both sets are partially answered during the execution of the ExaFlooding RD activities, part (or parts) of them are answered. There is a need to redefine the requirements space of the requested resource based on the concept of the vector. Redefining the requirements space of the requested resource based on the concept of vector enables the ExaFlooding RD to be able to decide on the response to the request (s) related to the request by the constituent elements of the global activity.

The ExaFlooding RD defines four $\text{Process}_{\text{Requirement}_{IO}}$, $\text{Process}_{\text{Requirement}_{file}}$, $\text{Process}_{\text{Requirement}_{\text{memory}}}$, and $\text{Process}_{\text{Requirement}_{process}}$ vectors for the required resource space. The direction of these four vectors in the direction of the vector is one and the size of each vector indicates the amount of need for the source specified by the vector. Any computational element that can satisfy a part (or parts) of these four vectors (or any single vector) reduces the size of the vector. From the point of view of the flooding resource discovery element, each computational element responding to the part (s) of the request creates a $\text{Process}_{Answer_x}$ *a* vector in the direction of the negative vector whose size is equal to the amount of resource that can be allocated to the request.

The RAS space is the central space of activities related to the element of flooding resource management. This space is considered as the space that defines the space requirements of the process. In the ideal state, there are four maps between the RAS space and the four $\text{Process}_{\text{Requirement}_X}$ vectors. These four maps transfer the constituent elements of the RAS space to the four $\text{Process}_{\text{Requirement}_X}$ quadratic vectors. $\text{Process}_{\text{Requirement}_X}$ vector space mapping and RAS space constituent members do not normally follow the corresponding mapping pattern. In distributed large-scale systems, each member of the RAS space is typically mapped to more than one $\text{Process}_{\text{Requirement}_X}$ vector space. This is because a request attribute may have a requirement in any of the four dimensions of the $\text{Process}_{\text{Requirement}_X}$ vector and four-vectors need to be considered to respond to it.

The occurrence of a dynamic and interactive event may cause the set of elements that make up the RAS space in the requesting process to change if the ExaFlooding RD mechanism lacks a mechanism for reconciling the RAS defined in the element-activating process flooding resource discovery and RAS used by the ExaFlooding RD, in which case it may be between process request properties and the properties requested by the ExaFlooding RD due to the occurrence of a dynamic and interactive event and its impact on Process request properties may differ.

The occurrence of a dynamic and interactive event may prevent the mapping between the $\text{Process}_{\text{Requirement}_X}$ and RAS vector spaces used by the ExaFlooding RD. Before the dynamic and interactive event, all the properties described in the RAS request were mapped to the $\text{Process}_{\text{Requirement}_X}$ vector spaces. Before a dynamic and interactive event occurs, each $\text{Process}_{\text{Requirement}_X}$ vector is answered by one (or more) $\text{Process}_{Answer_x}$ vector spaces. The occurrence of a dynamic and interactive event causes $\text{Process}_{Answer_x}$ vector spaces to be unable to respond to $\text{Process}_{\text{Requirement}_X}$ vector spaces due to changes in request properties or changing the RAS mapping pattern to $\text{Process}_{\text{Requirement}_X}$ spaces. This is due to a change in the pattern describing $\text{Process}_{\text{Requirement}_X}$ vector spaces or a change in RAS. Due to the consideration of the RAS Checker unit in the framework architecture of the ExaFlooding RD, the changes made to RAS apply the flooding resource activation management element process to the RAS used by the ExaFlooding RD. Therefore, the main challenge of the inability of $\text{Process}_{\text{Requirement}_X}$ vector spaces to respond by $\text{Process}_{Answer_x}$ vector spaces go back to the concept of changing the RAS mapping pattern to $\text{Process}_{\text{Requirement}_X}$ vector spaces.

Two scenarios can be used to solve this challenge. In the first scenario, the resource discovery activity in the computational element is canceled, and based on the new $\text{Process}_{\text{Requirement}_X}$ state, if the computational element can respond to the $\text{Process}_{\text{Requirement}_X}$ vector spaces by the $\text{Process}_{Answer_x}$ vector spaces, then the relationship between the two vector spaces mentioned is defined again. In the second scenario, the concept of $\text{Process}_{\text{Requirement}_X}$ virtual vector space is used. In this scenario, for $\text{Process}_{Answer_x}$ vector space, a vector $\text{Process}_{\text{Requirement}_X}$ is created, which is the result of mapping the requests in several $\text{Process}_{\text{Requirement}_X}$ vector spaces and by $\text{Process}_{Answer_x}$ vector space. It is responsive. From the point of view of the ExaFlooding RD, in such a situation the computational element in which the dynamic and interactive event has taken place, has the possibility and ability to respond to a part of the request. In the virtual vector creation scenario, the ExaFlooding RD, given that the axis of the function of the ExaFlooding RD is on-demand, and the ExaFlooding RD, based on the concept of global activity, has created a structure to respond to the request. Donors are parts of the request that can be answered by the current structure.

When a dynamic and interactive event occurs, in the Alpha computational element responding to the part (s) of the processing request, both the request vector and the request-response vector are affected by the event. Be dynamic and interactive. In this case, the $\text{Process}_{\text{Requirement}_X}$ *a* vector may have an angle with the $\text{Process}_{Answer_x}$ vector. The occurrence of a dynamic and interactive event may cause the angle between the two $\text{Process}_{\text{Requirement}_X}$ vectors and the $\text{Process}_{Answer_x}$ vectors to mean the ability to respond to part of the request process requirement. In such a situation, the ExaFlooding RD creates a virtual vector $\text{Process}_{\text{Requirement}_X}$ using the Virtual Vector element shown in the framework shown in Fig. 5. The reason for the virtualization of the $\text{Process}_{\text{Requirement}_X}$, is that in the structure related to responding to the RAS request of the ExaFlooding RD, it is not possible to specify the created response structure of the computational element that needs The values presented by the ExaFlooding RD must be equal to $\text{Process}_{\text{Requirement}_X}$ and answered by the $\text{Process}_{Answer_x}$*a* vector in that computational element. The ExaFlooding RD, in the Alpha computational element, defines the corresponding virtual vector when a dynamic and interactive event occurs to manage the angle between the $\text{Process}_{\text{Requirement}_X}$ vector and the $\text{Process}_{Answer_x}$ vector. If the occurrence of a dynamic and interactive event in the Alpha computational element is such that the outcome of the events in the Alpha computational element does not create an angle between the $\text{Process}_{\text{Requirement}_X}$ vector and the $\text{Process}_{Answer_x}$ vector, then The ExaFlooding RD does not use the concept of a virtual vector and implements the source discovery process based on the response structure pattern.

The difference between the $\text{Process}_{\text{Requirement}_X}$ vector and the $\text{Process}_{\text{Requirement}_X}$ vector represents the parts of the request that cannot be answered due to a dynamic and interactive event and cannot be answered. Made a space between the process vectors $\text{Process}_{Answer_x}$ and the part of RAS that corresponds to the difference between the two vectors mentioned. The vector $\text{Process}_{\text{Requirement}_X}$ is the result of the response of several $\text{Process}_{\text{Requirement}_X}$ vector spaces by the $\text{Process}_{Answer_x}$ vector space, so the vector $\text{Process}_{\text{Requirement}_X}$ represents the set of activities that the ExaFlooding RD has done to respond to the request by the response structure. On the other hand, by mapping each part of the request that is about resource type x to the $\text{Process}_{\text{Requirement}_X}$ the process when calling the ExaFlooding RD, whenever a dynamic and interactive event occurs in the Alpha computational element Given the difference of the initial vector related to the mapping of RAS requirements related to the process in terms of the type of source x from the virtual vector created at the moment of the dynamic and interactive event, sections can be extracted from Regarding the type of resource specified, RAS x stated that the current account structure created by the ExaFlooding RD cannot respond to them and should be re-accounted for by the ExaFlooding RD. If the ExaFlooding RD needs to define a virtual vector to manage the dynamic and interactive event in the Alpha computational element and to form an angle between the $\text{Process}_{\text{Requirement}_X}$ vector and the $\text{Process}_{Answer_x}$ vector, then according to the change The computational elements of the Q System member, function number 12, or the function that describes the function of the ExaFlooding RD, are changed. Changing the computational elements of the Q System member as well as the occurrence of a dynamic and interactive event requires the activation of the RAS Checker unit and redefining the RAS related to the request according to the information of the time and space constraints unit and D&I and redefining the Q System membership condition is.

5 Evaluation

To evaluate the ExaFlooding RD framework, in Distributed Exascale Systems, a computationally distributed peer-to-peer system has been used [22, 31]. In this computational system, the system manager typically uses the functional concept based on the flood method to find the source of the requested process in the extended system. In the computing system [22, 31], when a request occurs that the local computing element cannot respond to, due to the lack of central structures as well as pre-defined response structures of the concept of flooding resource discovery based on the version the specific defined in this system is used. On the other hand, in the computational system [22, 31], the concept of resource classification is based on the classification used by the operating system and based on four types of file sources: input/output, memory, and processing. Using such a template to manage and adapt it to the pattern used in the ExaFlooding RD framework to classify resources allows the system to be used to evaluate the ExaFlooding RD framework.

To evaluate the ExaFlooding RD framework, the system [22, 31] has implemented two types of global activities corresponding to the implementation of two scientific and applied programs. Charm ++ [34] and WRF [35] are considered as two software that require high performance computing and an extensive processing system to evaluate the ExaFlooding RD framework. Each of these two software, based on global activity, either uses the computational resources available in the system or invokes the unstructured resource discovery defined in the system [31] to find the resource it needs. To. Therefore, at any point in the process of implementing system activities, two types of global activities are running in the computing system. Each member of the system can play a role in the implementation of one or two activities at any time.

The number of elements of the computational system is equal to 180 computational elements. The composition of the computing system from 180 computing elements makes it possible to consider the computing system [31] considered as a test platform as a comprehensive system for each of the two software. The software mentioned is normally running on a smaller number of computing elements, so running them on 180 computing elements allows the status of the software when running on a large system. Are, analyzed, and examined.

To create dynamic and interactive events, in Computational Element 6, a special version of system management software [31] is used in which the resource discovery uses the ExaFlooding RD framework. The reason for choosing Computational Element No. 6 is since in most cases of implementation of the mentioned scientific and applied programs, the process requesting the unstructured resource discovery is executed in this computational element and the management element is called. Discover the source used in the computational system [31]. In the process of implementing the scientific and practical programs mentioned in the computational system used as a test platform, other computational elements such as computational element number 6 can also be considered. In this type of computational element, the absence of one (or more than one) resource requested by one (or more) processes causes the management element related to the local computational element to call the resource discovery based on the expression mechanism in [22].

Computational element 6, the system process manager has been changed. This enables the system manager in this computational element to manage the three modes of process creation, communication, and interaction with the system environment that lead to a dynamic and interactive event.

The computational element No. 6 can manage situations that lead to dynamic and interactive events occurring in processes related to global activity. Figure 6 shows the normal, dynamic, and interactive event occurrence status, as well as the number of events, responded by ExaFlooding RD.

As can be seen in Fig. 6, on average, 50 requests occur in the 50 times of testing in Computational Element 6, leading to the call of the resource discovery. From the point of view of the management element of Computational System No. 6, this issue is related to the concept that normally in the implementation of scientific and applied programs by Computational System No. 6, at each execution, 15 requests for access to the resource by the process (or Processes) that the computing system designed to run a scientific and applied program cannot respond to. By examining the process of implementing activities related to resource discovery, the status of global activities that the process (or processes) in the computational element No. 6 have called the resource management element and also investigating the occurrence of the event Dynamic and interactive in the process of performing resource discovery activities, an average of 9 dynamic and interactive events occur for each test run. These 9 dynamic and interactive events are both dynamic and interactive events that change the status of the process (or processes) that activate the resource management element and include dynamic and interactive events that affect global activity created by the resource discovery affects or affects the element containing resource. Occurrence of 9 dynamic and interactive events on average in the process of carrying out the activities related to discovering the source required for the process (or processes) of computational element No. 6, means that in an average of 58% of cases Activities related to the resource management element occur as a dynamic and interactive event, the lack of which causes the process of implementing resource discovery activities to fail. The occurrence of a dynamic and interactive event in more than half of the cases leading to the call of resource discovery in the Distributed Exascale Systems necessitates the definition of a framework that can handle the unstructured resource discovery mechanism used in the system. [31] develop and be able to manage dynamic and interactive event situations.

As can be seen in Fig. 6, on average, the ExaFlooding RD, after a dynamic and interactive event and based on the framework introduced in Fig. 5, out of 9 dynamic and interactive events. On the activities of the resource management element, it can manage 5 events and prevent the failure of the activities related to the

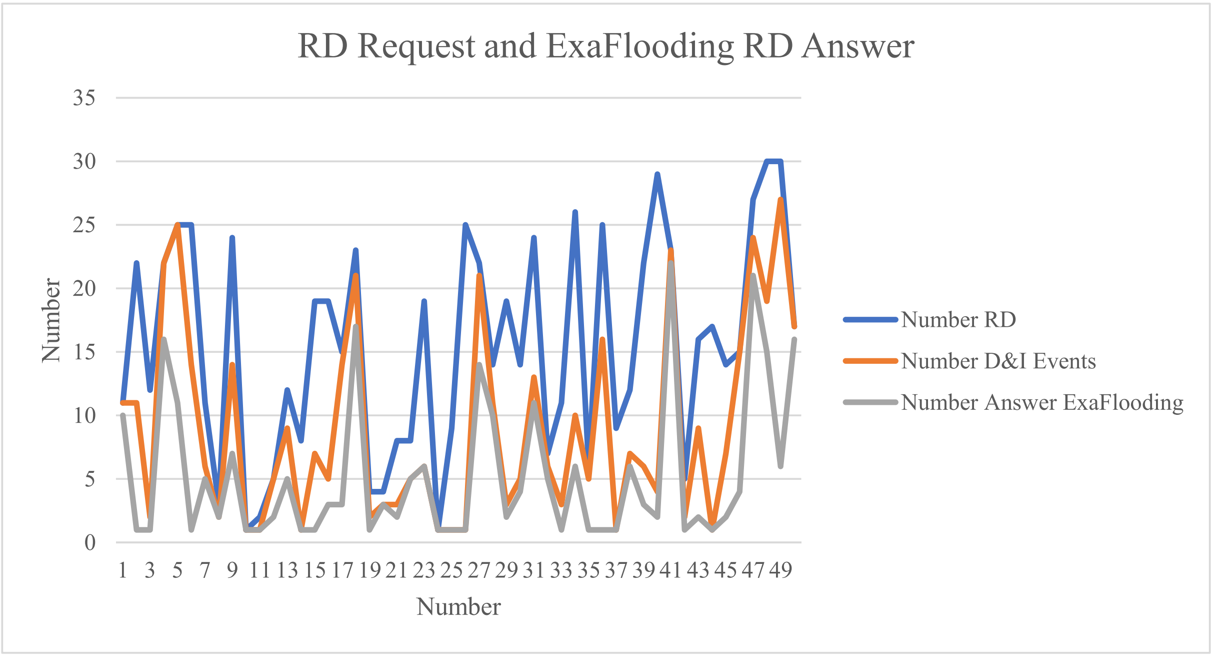

Fig. 6 Number of requests leading to resource discovery call, number of dynamic and interactive events discovered in the resource discovery process by ExaFlooding RD, and number of dynamic and interactive events managed by ExaFlooding RD.

unstructured resource discovery. Management 5 events affecting the activities of resource discovery by the ExaFlooding RD, means that the elements of the framework introduced in Fig. 5 or are able to develop the concept of request and use formula 7, Considering the changes made to the RAS concept by the RAS Checker element and activating the Review Engine element and defining the new RAS based on the occurrence of dynamic and interactive events, constraints and time and space constraints governing the request as well as Q System status, or by using the concept of virtual vector and activating the corresponding element in the framework shown in Fig. 5, manages dynamic and interactive events and by changing the response structure based on the new situation, prevents Failure of the resource discovery activities by examining dynamic and interactive events that can be managed by the ExaFlooding RD, based on the framework introduced in Fig. 5, it can be concluded that in 59% of cases, the framework provided the ability to manage dynamic events and It has an effective interaction on the activities of the unstructured flooding resource discovery used in the system [31].

Table 1 shows the relationship between the number of dynamic and interactive events affecting the function of the flooding resource discovery as well as the number of events managed by the ExaFlooding RD.

As can be seen in Table 1, by considering the variable number of occurrences of dynamic and interactive events affecting the function of the management element, discover the flooding resource as an independent variable and consider the number of times that the management element.

ExaFlooding RD can manage the effects of the mentioned dynamic and interactive events on the function of flooding resource discovery as a dependent variable, it is determined that the dependent variable has a strong dependence on the independent variable. The number 0.6 indicates that if more experiments are repeated and the experiment is performed for a sufficient number of cases, it can be stated that in 59 to 60% of cases, the ExaFlooding RD can manage dynamic events and it has an effective interaction on the function of the flooding resource discovery. As shown in Fig. 5, in the framework of the ExaFlooding RD, the focus of the framework is based on the D&I concept, and in the event of a D&I event, the system manager is required to perform related activities. Flooding resource discovery based on the traditional mechanism continues to carry out activities related to resource discovery based on the ExaFlooding RD. This causes the ExaFlooding RD call to be completely dependent on a dynamic and interactive event occurring (by changing the status of the

Table 1 The relationship between the number of dynamic and interactive events that occur and the number of events managed by the ExaFlooding RD management element

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.785[a]	.616	.608	3.56497

a. Predictors: (Constant), D&I Events

trigger process, the created global activity state by the resource discovery, or the element containing discovered resource).

Experiments were performed and, according to Table 1, the ExaFlooding RD was able to manage 60% of the dynamic and interactive events affecting the function of the resource discovery in a situation where the number of experiments increases. The reason for the inability of this mechanism to respond to all dynamic and interactive events affecting the function of the resource discovery, can be due to a) the mechanism used to detect the occurrence of dynamic and interactive events used in Fig. 5, b) The ability to redefine the RAS system based on the Q System status, the temporal and spatial constraints on responding to the request and the effects of a dynamic and interactive event c) The ability to redefine the Q System as a result of the event Dynamic and interactive event d) Pattern used for similarity function e) Process status of the resource discovery after the dynamic and interactive event occurs, and) Possibility of defining the virtual vector after the event be a dynamic and interactive event. Each of the above factors can lead to the inability of the system review element to create RAS and define the Q System and, consequently, the possibility of continuing to perform activities related to the resource management element.

In experiments, the similarity function is considered on the centrality of the computational element, this has caused that in the $QM_{i,}$ matrix, the centrality of creating the matrix was based on the type of computational source. This causes the similarity coefficient in high numerical experiments to be about 95%. Considering this degree of similarity causes the result of executing formula 13 and finding the elements of the matrix RM_{ij} is equivalent to finding the elements of the matrix, RM_{ij} is not considering the concept of similarity and finding the source without considering the concept of the matching function. In the function of the ExaFlooding RD, due to the definition of the similarity element in the framework shown in Fig. 5, and considering formula 13 and the possibility of defining the $QM_{i,}$ matrix, in terms of one or all of the sources defined in the computing system based on each P or degree of similarity in the formula, it is possible for the ExaFlooding RD to increase its response rate to its maximum. The ExaFlooding RD can increase the percentage of dynamic and interactive event management affecting the function of the resource discovery by considering different values for P in Formula 13. However, considering lower values for P in Formula 13 increases the research space of the resource management element and increases the likelihood of finding a resource that has a degree of similarity P with the requested resource. However, this may cause the resource found to be inconsistent with the RAS described by the requesting process. Examining the experiments, it was found that the use of function number 13 and changing the value of P, in situations where the occurrence of a dynamic and interactive event has an effect on the activating process of the resource discovery and consequently change RAS, can be used as a suitable solution. In this case, the RAS revision provides the ability for the resource discovery to adapt to changes made to the RAS after a dynamic and interactive event has occurred relative to the original RAS created by the resource management element activator process define the degree of similarity.

Figure 7 shows the dynamic and interactive events detected by the ExaRD framework [21] and the ExaFlooding RD framework that failed resource discovery activities.

In addition to the ExaFlooding RD, the ExaRD [21] is used to perform the experiment shown in Fig. 7 in the computing system used for the experiment. The ExaRD Resource Discovery Manager element manages resource discovery activities in a general way, without considering the unique features of the unstructured resource discovery process. The ExaRD, like the ExaFlooding RD, can manage dynamic and interactive events affecting the activities of the resource

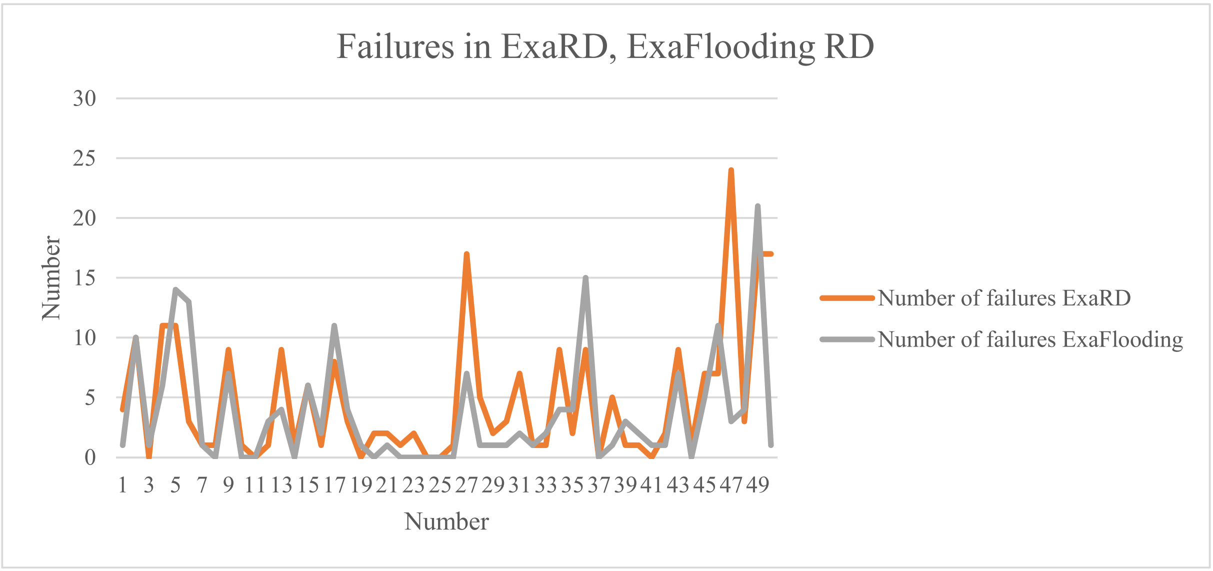

Fig. 7 Number of dynamic and interactive events leading to failure of the resource discovery

management element. The ExaRD, by redefining the concept of the central element of performing resource discovery activities from source finding to the concepts of request state and process state, can occur in the event of a dynamic and interactive event in the request process. Based on the information collected during the source discovery process, the resource provider manages a dynamic and interactive event and responds to a dynamic and interactive request. As can be seen in the experiment shown in Fig. 7, on average, 50 times the test was performed in computational element 6 based on the use of the ExaRD as the resource discovery, in 5 times the ExaRD resource discovery. It cannot manage dynamic and interactive events affecting the functioning of the resource discovery. By repeating the experiments in computational element number 6 and using the management element ExaFlooding RD as the resource discovery, it is determined that on average, the experiment cannot manage 4 dynamic and interactive events affecting the implementation of resource discovery activities. By experimenting to evaluate the ability of dynamic and interactive event management and the continuation of the process of activities related to resource discovery, it was determined that on average, the ExaRD can manage 4 dynamic and interactive events and continue the implementation process. Resource discovery and, on average, the ExaFlooding RD can manage 5 dynamic and interactive events affecting the function of the resource management element and continue the process of implementing resource discovery activities.

Execution of the test, both in the form of inability to manage dynamic and interactive events affecting the activity of the resource management element or in the form of dynamic and interactive event management affecting the activity of the resource management element, indicates that the ExaFlooding RD has capabilities equivalent to the capabilities of the ExaRD and in some cases improves or does not detect dynamic and interactive events affecting the resource management element. The difference between the ExaFlooding RD and the ExaRD in some cases is due to the redefinition of the application concept based on Formula 7 and the consideration of the framework shown in Fig. 2 for the application. The ExaRD framework considers the concept of request based on RNS and the concept of request requirements based on Request Image. This is while in the framework of ExaFlooding RD, the concept of request is described as a pivotal and developed concept based on Formula 7, and from the point of view of the ExaFlooding RD, the framework shown in Fig. 2 makes the concept Request and RAS The descriptor of the request should be rewritten according to Formula 8, taking into account the system conditions, the process and the process of carrying out the activities related to the discovery of the source. The development of the concept of request in the ExaFlooding RD makes the difference between the two frameworks in determining or not determining the dynamic and interactive events that govern the resource management element.

Figure 8 shows the number of dynamic and interactive events that affect the functioning of the resource discovery and are managed and accounted for by the ExaFlooding RD in the Cactus [31, 36] frameworks.

As can be seen in Fig. 8, the ExaFlooding RD resource discovery, in addition to being implemented and evaluated in a distributed Exascale framework-based computing system [31] is a dynamic and interactive framework-based computing system [36] has also been implemented and evaluated. In the Cactus framework, the properties of the resources contained in the elements of the computing system change over time.

The cactus framework includes most of the same concepts of the system described in related work. This framework includes the majority of the patterns in the operation management field that is related to the unstructured flooding resource discovery. The cactus framework includes most of the concepts expressed in the system discussed in related work. This makes the cactus framework able to be used as the main framework to evaluate the presented framework to manage dynamic and interactive events on the performance of the unstructured flooding resource discovery. The PMamut framework also includes the mechanisms that are used to create responsive structures based on the concept of unstructured scalability, mainly flooding for the management of dynamic and interactive events.

Using the two mentioned frameworks and implementing the proposed framework for managing dynamic and interactive events affecting the functionality of the unstructured flooding resource discovery provides the capability to assess how the framework works based on previously defined mechanisms and patterns of operation of the unstructured flooding resource discovery as well as the functionality of the framework in a situation where the unstructured resource discovery is used to make a responsive structure.

Regarding the evaluation of the proposed framework based on the Cactus and PMamut, considering that the main goal is to determine whether the proposed framework has the capability of managing the dynamic and interactive events affecting the function of the unstructured flooding resource discovery. Therefore, the main activities related to the evaluation are based on how to prevent the failure of the operation related to the unstructured flooding resource discovery.

In distributed Exascale computing systems, the dynamic and interactive events change the status of the system descriptor parameters from the perspective of the resource discovery (like any other element of the computational system management) in a way that the unstructured flooding resource discovery cannot manage the mentioned situation. In such a case, the computational system is failing the process of implementing activities related to the application, and considering other parameters regarding the functionality of a framework that prevents the failure of the execution of the activities of the resource discovery is almost unavoidable.

One of the most important features of the Cactus framework, which provides the ability to implement

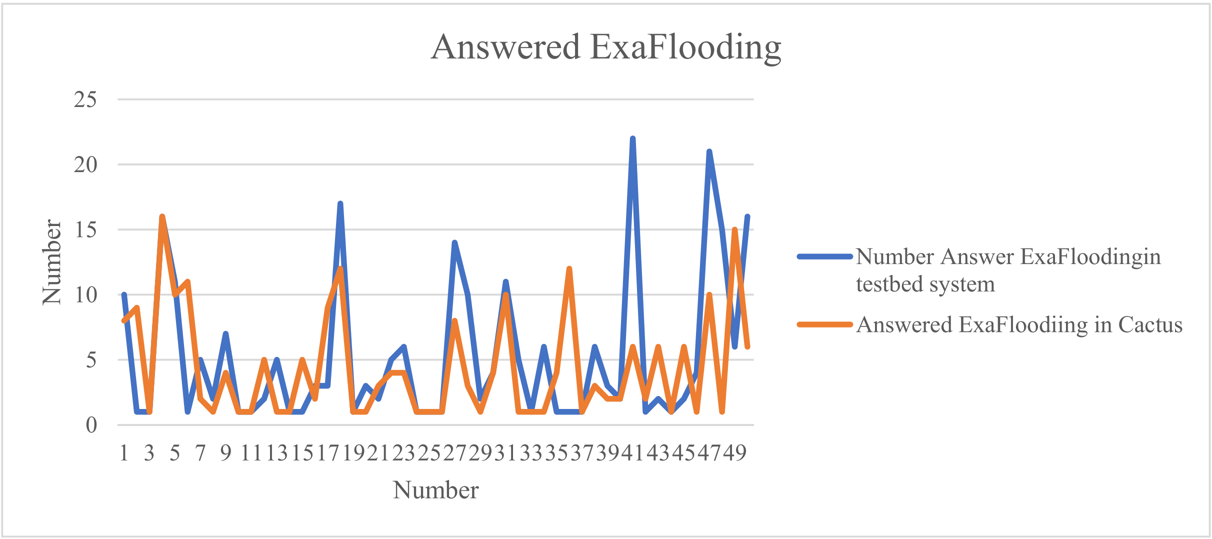

Fig. 8 Number of dynamic and interactive events affecting the function of resource discovery responded by the ExaFlooding RD in frames [31, 36]

Table 2 Dependence between the number of dynamic and interactive events affecting the function of the resource discovery and the number of events managed by the ExaFlooding RD in the system

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.779a	.607	.598	2.61464

a. Predictors: (Constant), D&I Events

the ExaFlooding RD, is its open resource and versatile nature, which requires parallel computing to solve scientific and engineering problems. In the Cactus framework, there is a flexible structure due to the resource variability defined in the computational elements of the system member, to create widely distributed Exascale computing systems. The mechanism used to select resources in the Cactus framework is centered on the fact that if the efficiency of the system is reduced, the pattern of resource allocation to computational processes can be changed. The architecture of the Cactus framework is based on the resource management element, the process migration, and the resource allocation management element. The resource allocation management element in the Cactus framework uses a matchmaking algorithm that can select the most appropriate resource as the result of the requesting process. In the matchmaking algorithm, the management element manages the allocation of process requests based on the operating system version, minimum memory, and minimum bandwidth.

As can be seen in Fig. 8, the results of 50 experiments performed on Computational Element 6 show that the ExaFlooding RD in the system [31] is capable of managing more dynamic events and interaction is effective on the function of the resource discovery. In the diagram shown in Fig. 8, this is especially evident in the high repetition of the experiment and when the number of times the test increases. On the other hand, examining the experiments shows that the mechanism of ExaFlooding RD in both frameworks [31, 36] on average in each test run can manage 5 dynamic and interactive events affecting the function of the management element is resource discovery. This means that in the average state and taking into account all system states, whether in system states where the number of test runs is low or in situations where the number of test runs increases the functionality of the ExaFlooding RD follows almost the same pattern in both frameworks. From this approach, it can be stated that if the tests are repeated for a large number and the system is in equilibrium, the function of the ExaFlooding RD does not depend on the specific features of the system and is based on the function expressed. It works in formula number 12. The dependence of the function of the ExaFlooding RD on the function number 12 and considering even a few machine sentences cause the ExaFlooding RD to perform exploration activities based on the ability model to normalize the RAS part or parts by the computational resource element. The definition of the request in the function of the ExaFlooding RD is not dependent on the system in which it performs the source discovery activities, and is based on the framework shown in Fig. 2 and redefining the concepts related to the request based on the number formulas 8 (in order to consider the system conditions, constraints and limitations as well as the process of implementing resource discovery activities in RAS and not considering the concept of RAS as an abstract concept defined by the process), 9 (in order to consider Concept of Process Plan Considering the conditions governing the process in the large-scale computational system distributed on the concept of RAS), 12 (redefining the concept of Q System based on the concept of RAS) and also 11 (in order to consider the beneficiaries on the applicant process and process implementation of resource discovery activities) and makes the concept of RAS in the function of the ExaFlooding RD as a central concept and independent of the characteristics of the system in which the resource discovery activity is performed.

Given this, it is expected that between the number of dynamic and interactive events affecting the function of the resource discovery and the number of events managed by the ExaFlooding RD in the model system similar to that shown in Table 1, be established. Table 2 shows the relationship between the number of dynamic and interactive events affecting the function of the resource discovery and the number of events managed by the ExaFlooding RD in the system.

As shown in Table 2, the dependence between two independent variables and the dependency shown in

Table 2 follows the same pattern as the dependency shown in Table 1.

Redefining the concept of demand and considering the framework shown in Fig. 2 for the concept of demand as well as managing activity based on RAS revisions due to changes in D&I, changes in global activity, and the Q System, as well as changes in occurrence. Given in the applicant process is based on the framework shown in Fig. 5.

6 Discussions

Resource discovery is considered as one of the main mechanisms used to manage dynamic and interactive events occurring in programs running on Distributed Exascale Systems. This enables the function of the resource discovery in Distributed Exascale Systems, in addition to the task of finding the resource, as a tool for creating response structures that lead to the management of dynamic and interactive events is considered. The nature of dynamic and interactive events in Exascale systems is distributed in such a way that at the moment of the event due to changes in the status of the system descriptor parameters and the position of the system descriptor parameters in the state Lack of defined and known for the constituent elements of the system manager is not precise, specific and definite. For dynamic and interactive event management in which the status of system descriptor parameters is not clear to the constituent elements of the system management element, the resource management element must be able to rely on a set of parameters such as mechanisms. Traditional unstructured resource discovery creates a responsive structure for dynamic and interactive events. From this approach, traditional unstructured resource discovery mechanisms can be used as a tool to create responsive structures for dynamic and interactive event management. Special functional features of the unstructured resource management element, such as the lack of definition of the limitations and constraints of the research space, can be considered as features for the successful management of dynamic and interactive events. On the other hand, in the management structure of unstructured resource discovery, the focus of the functionality function of this element is based on the concept of RAS as well as the similarity of the resource found with the requested resource. The centrality of these two features makes it possible to discover an unstructured source from the management element for the management of events due to which only several variables describing the state of the system can describe the known state for the management element. The system represents and the main goal is to create a structure that responds most closely to the requests created to respond to the dynamic and interactive event. In Distributed Exascale Systems, the nature of the dynamic and interactive event is such that in the event of a dynamic and interactive event, the main purpose of the system manager is to respond to the situation created by the dynamic event. And the interaction and requests created are the result of a dynamic and interactive event. In such a situation, if the resource discovery can create a response structure with the maximum possible resemblance to the situation, then it can manage dynamic and interactive events. In this paper, by defining the functionality function of the unstructured source discovery, it was determined that the functionality function of this element operates on the centrality of the concept of similarity. In such a situation, the occurrence of a dynamic and interactive event is considered as an activator of the management element of unstructured resource discovery.

The model governing the function of the unstructured resource discovery and the implementation of activities related to the said element outside the system and not using pre-defined structures to manage the request-response process leads to the activation of the resource management element and by another possibility of dynamic and interactive event occurrence in a distributed Exascale system that also affects the function of the unstructured resource discovery is the need to develop the concept of the function of the unstructured resource management element to manage dynamic events and effective interaction on the function of this element is essential.

In this paper, to develop the concept of the function of unstructured resource discovery based on the flood method, centrality based on the concept of request has been considered. For this purpose, based on Fig. 2, this paper introduces a framework for the concept of demand in Distributed Exascale Systems. This framework is more developed than the concept of request activation of resource management element and includes the concept of global activity, dynamic and interactive event occurrence, beneficiaries on the process of activating resource management element as the process. The owner of the request and the more developed restrictions govern the process of responding to the request. In the context of ExaFlooding RD, given that one of the important features of the unstructured resource discovery is the concept of using the requested resource similarity to the request described in RAS, so unlike the structured resource discovery, the element discovered by the resource discovery may not be a single element. This is described in the function of the traditional unstructured resource management element function as well as the function of the ExaFlooding RD resource management element function, using the concept of normalizing minimal polynomials. In this paper, the use of the concept of normalization of request descriptive polynomials expressed by RAS as well as resource descriptive polynomials and the possibility of normalizing part of a minimal RAS polynomial by a computational element may cause more than creating an element or structure to respond to RAS. The concept of normalizing minimal RAS polynomials by polynomials related to each source makes the ExaFlooding RD resource discovery a concept of the Q System or a system in which resources can normalize part of a minimal polynomial. RAS is used. In this paper, the concept of the Q System is considered as a function of time and event. Considering the concept of Q System based on the above-mentioned independent variables makes it possible for a dynamic and interactive event to occur and affect each element of the Q System in a way that cannot normalize RAS polynomials. It is possible to change Q System members. The dynamics of the Q System enable the ExaFlooding RD resource discovery while managing the process of creating a response structure, to be able to manage changes that have occurred to previously discovered resources that are part of the response structure process be a dynamic and interactive event.

In this paper, like traditional computing systems, the central concept of the request is based on the description of the request by RAS, but with the difference that RAS is intended to design the management element of ExaFlooding RD, using the mathematical model of the effect of discrete independent variables on each other. Considering the constraints and limitations governing the request and the response process to the request, as well as the concept of the effects of dynamic and interactive events affecting the functioning of the resource management element on the RAS concept described by the processor. This means that the RAS considered in the ExaFlooding RD, unlike traditional computing systems, is not considered as an abstract concept described by the process and includes the mentioned concepts. The initial development of the RAS concept from the abstract concept described by the process to the concept that includes the main elements influencing the description of RAS in the framework introduced in the paper leads to the search process of the ExaFlooding RD compared to the resource management element. ExaRD uses the abstract RAS described by the process, to be done more accurately and in a more specific range. On the other hand, changing the definition of RAS from the abstract state described by the processor to the developed state in the framework expressed in this paper makes the concept of similarity and the creation of similarity-based response structures can be considered by limiting factors.

In the developed framework for the concept of request in this paper, considering that in Distributed Exascale Systems, unlike traditional computing systems, the central element of the system is not considered the process, and The concept of global activity is considered as a central element of system management, so the concept of request in the ExaFlooding RD, in addition to considering the concepts related to the limitations of the request and constraints governing the process of responding to the request. Constraints on the occurrence of dynamic and interactive events include the effects of defining the process that activates the resource discovery on the global activity that is running in the computing system. Considering the mentioned concept for requesting and developing the concept of request based on the concept of global activity has caused in this article, the concept of process request plane and the ability plane to respond to the requests of the resource owner. Using the concept of global activity and considering it in the concept of request causes the ExaFlooding RD to define the affine plane of the process for each process activating the ExaFlooding RD. The process requirements at any point in the resource discovery process are defined by the ExaFlooding RD on the affine request plane, and any changes to the delivery plane request are specific and specific about payment requests. The affine plane created for the process includes both the global activity of which the process is a part and the process of responding to the request of the process activating the resource management element by the ExaFlooding RD. On the other hand, like the affine plane defined for the process, the response affine plane can be defined, which indicates the capabilities of the process and the resources that the process owns and can respond to the request for the process Other. The use of the affine plane pattern designed for the process based on the concept of global activity and the transfer of the request or the ability to respond to the request based on the information contained in the affine plane of the process makes it possible to consider changes in requests and process capability. It is also possible to check the response of the request by the processor. Accordingly, if the processing request plane intersects with the resource plane or other process capability, it will be possible to respond to the processing request by one (or more) resources. In this paper, the processing plane in the Exascale system is designed in the form of a three-dimensional space that allows the descriptive features of the request to be scalar and the resource capability to the vector, as well as the possibility of defining different mechanisms. Response to process request activation provides the ExaFlooding RD resource discovery. This means that in the context of ExaFlooding RD, the concept of page layout is not only based on requirements and capabilities, but also on the characteristics of the mechanism used to discover the resource.

In the presented framework, in addition to considering the concept of global activity, the status of the system, as well as the elements affecting the process, are also discussed as factors that should be considered in describing the request. Considering the effects of system status and the elements affecting the process makes it possible to consider the effects of these changes in the concept of process request in case of changes in system status for any reason. In the context of ExaFlooding RD, a four-dimensional space is used to consider changes in system status based on the concept of request, and then the feasibility of defining RAS in the new state of the system is investigated using the concept of partial derivation. This allows ExaFlooding RD to decide on the feasibility of responding in the system in which the resource is found as well as the request is defined. Using the mathematical function presented in this paper to investigate the feasibility of defining RAS in the new state of the system can be used as the ExaFlooding RD resource discovery to detect the process requests described by RAS after an event has occurred. Dynamic and interactive as well as changing the state of the system, it is not possible to respond to them, help. On the other hand, in this paper, the definition of the system includes four main spaces according to which it is possible to accurately describe the operation of the distributed Exascale system based on the concept of global activity, properties of computational elements, properties of demand and it also showed the concept of time. In the formula presented in this paper, the ExaFlooding RD can determine what properties change in the system state, depending on which of the independent variables defining the system lacks definition. As a result of a dynamic and interactive event or a change in the state of global activity, the RAS cannot make decisions.

In a distributed Exascale system, given that at any point in the process of implementing a scientific program, there is a possibility of a dynamic and interactive event occurring, and a dynamic and interactive event may occur. The computational process is effective in that this computational process is interacting with the activation process of the resource discovery, so to manage changes in other processes and their impact on the requesting process in the management element ExaFlooding RD uses the concept of an inter-process plane. In the ExaFlooding RD management element, to check the communication and interactive space of the process space activating the resource discovery and the beneficiary processes with this process, the linear combination result between the stake processes and the resource activating element activating process is used. Be. Using the concept of linear composition enables the ExaFlooding RD to generate an affine plane related to inter-process communications and interactions to examine the interactions and connections between processes. The activator of the resource discovery and any other process is more beneficial than the said process and can calculate the linear result of the effects of the stakeholder process on the resource activator of the resource discovery at any time during the implementation of the scientific program.

The ExaFlooding RD framework presented in this paper focuses on reviewing the occurrence of each event affecting the system, dynamic and interactive request, as well as recognizing the effects of dynamic and interactive event occurrence on request and request-based review. It is dynamic and interactive, taking into account the constraints and limitations of the request and the process of responding to the request. Introduced framework for the ExaFlooding RD resource discovery, considering that any change in the listed variables may affect the entire response structure, including the elements discovered and the elements it is discovering, or the request. Giving each of the events leading up to the system review, taking into account the effects of the dynamic and interactive event as well as the constraints governing the request and the process of responding to the request, reviews the RAS as well as the Q System. By redefining the function of the resource discovery based on the conditions of the distribution within the ExaFlooding RD framework, the RAS and Q System revision provides the capability Exascale system for the ExaFlooding RD to create a scenario by classifying response structure based on the concept of ExaFlooding RD functionality function, analyze the effects of dynamic and interactive event occurrence on it and based on the concept of virtual vector describing the response structure or the concept of developed similarity function dynamic and interactive event affecting the function of the resource discovery element.

7 Conclusion

In this paper, the ExaFlooding RD unstructured resource discovery is used to redefine dynamic and interactive events affecting the unstructured resource discovery process in Distributed Exascale Systems, based on redefining the application concept as well as the functionality function of the unstructured resource discovery element. The framework provided for the ExaFlooding RD is based on the concept of redefining the request and considering system-related features, dynamic and interactive events, the request plane, and the capabilities of the processes involved in the resource discovery process, including the request process. The interaction and communication of other processes in the system are designed after the dynamic and interactive event occurs and the impact of the event on them, on the resource requesting process, and on the concept of the request. The focus of the ExaFlooding RD framework is on creating a response structure and reviewing the status of the response structure created based on the developed functionality function of the unstructured resource discovery in Distributed Exascale Systems after a dynamic and interactive event occurs. And its interaction and impact on resource discovery activities. Defining the Concept of functionality function of the Unstructured resource discovery and development for Distributed Exascale Systems provides the capability for the framework introduced for the ExaFlooding RD management element, which can be defined by defining the response structure and probability of impact. Dynamic and interactive data in two different scenarios for this structure, to manage dynamic and interactive events on the function of the resource discovery element. The framework introduced for the ExaFlooding RD management element is not dependent on the system in which it processes the requested resource, and its functionality is independent of system-specific features.

“Data sharing does not apply to this article as no datasets were generated or analyzed during the current study.”

Declarations

Conflict of Interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Khaneghah, E.M., Sharifi, M.: AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime. J. Supercomput. **67**(1), 1–30 (2014)
2. Adibi, E., Khaneghah, E.M.: Challenges of resource discovery to support distributed exascale computing environment. Azerbaijan Journal of High Performance Computing. **1**(2), 168–178 (2018)
3. Bidhendi, Z.E., Fakhri, P., Khaneghah, E.M.: Challenges of using unstructured P2P systems to support distributed Exascale computing. Azerbaijan Journal of High Performance Computing. **2**(1), 3–6 (2019)
4. Khaneghah, E. M., et al.: "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), (2018)
5. Burness, A.-L., et al.: Scalability evaluation of a distributed agent system. Distrib. Syst. Eng. **6.4**, 129 (1999)
6. Jogalekar, P., Woodside, M.: Evaluating the scalability of distributed systems. IEEE Trans. Parallel Distrib. Syst. **11**(6), 589–603 (2000)
7. Liu, X., Wang, S., Ji, H.: Double-layer P2P networks supporting semantic search and keeping scalability. Int. J. Commun. Syst. **27**(12), 3956–3970 (2014)
8. Senthuran, A., Hettiarachchi, S.: A review of dynamic scalability and dynamic scheduling in cloud-native distributed stream processing systems. ICDSMLA. **2020**, 1539–1553 (2019)
9. Kafhali, E., Said, et al.: Dynamic scalability model for containerized cloud services. Arab. J. Sci. Eng. **45**(12), 10693–10708 (2020)
10. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts Essentials. Wiley, Hoboken (2014)
11. Mousavi Khaneghah, E., Mirtaheri, S.L., Sharifi, M., Minaei Bidgoli, B.: Modeling and analysis of access transparency

and scalability in p2p distributed systems. Int. J. Commun. Syst. **27**(10), 2190–2214 (2014)

12. Khaneghah, E.M., et al.: The influence of exascale on resource discovery and defining an indicator. Azerbaijan Journal of High Performance Computing. **1**(1), 3–19 (2018)
13. Navimipour, N.J., et al.: Resource discovery mechanisms in grid systems: a survey. J. Netw. Comput. Appl. **41**, 389–410 (2014)
14. Bo, Jin. "Flooding-Based Resource Locating in Peer-to-Peer Networks." Electronics and Signal Processing. Springer, Berlin, Heidelberg, 2011. 671–678
15. Thampi, S. M.: "Survey of search and replication schemes in unstructured p2p networks." arXiv preprint arXiv: 1008.1629 (2010)
16. Khatibi, E., Sharifi, M.: Resource discovery mechanisms in pure unstructured peer-to-peer systems: a comprehensive survey. Peer-to-Peer Networking and Applications, 1–18 (2020)
17. Khatibi, E., et al.: "Dynamic multilevel feedback-based searching strategy in unstructured peer-to-peer systems." 2012 IEEE International Conference on Green Computing and Communications. IEEE, (2012)
18. Saeedvand, S., Aghdasi, H.S., Khanli, L.M.: Novel distributed dynamic backbone-based flooding in unstructured networks. Peer-to-Peer Networking and Applications. **13**(3), 872–889 (2020)
19. Bashmal, L., Almulifi, A., Kurdi, H.: Hybrid resource discovery algorithms for unstructured peer-to-peer networks. Procedia Computer Science. **109**, 289–296 (2017)
20. Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V., Haridi, S.: Peer-to-peer resource discovery in grids: models and systems. Futur. Gener. Comput. Syst. **23**(7), 864–878 (2007)
21. Adibi, E., Khaneghah, E. M.: "ExaRD: introducing a framework for empowerment of resource discovery to support distributed exascale computing systems with high consistency." Cluster Computing. **23**(4), 3349-3369 (2020)
22. Sharifi, M., Mirtaheri, S.L., Khaneghah, E.M.: A dynamic framework for integrated management of all types of resources in P2P systems. J. Supercomput. **52**(2), 149–170 (2010)
23. Schmid, S., Wattenhofer, R.: Structuring Unstructured Peer-to-Peer Networks. International Conference on High-Performance Computing, Springer, Berlin, Heidelberg (2007)
24. Ferretti, S.: "A general framework to analyze the fault-tolerance of unstructured P2P systems." 2010 Fourth UKSim European Symposium on Computer Modeling and Simulation. IEEE, (2010)
25. Oliveira, R., Bernardo, L., Pinto, P.: "Flooding techniques for resource discovery on high mobility MANETs." Workshop on Wireless Ad-hoc Networks. (2005)
26. Adibi, E., Khaneghah, E.M.: A mathematical model to describe resource discovery failure in distributed exascale computing systems. Peer-to-Peer Networking and Applications. **14**(3), 1021–1043 (2021)
27. Basukoski, A., et al.: Design and Implementation of a Hybrid P2P-Based Grid Resource Discovery System, pp. 119–128. Making Grids Work. Springer, Boston, MA (2008)
28. Alowayyed, S., Groen, D., Coveney, P.V., Hoekstra, A.G.: Multiscale computing in the exascale era. J. Computational Sci. **22**, 15–25 (2017)
29. Xiang, Q., Tony Wang, X., Jensen Zhang, J., Newman, H., Richard Yang, Y., Jace Liu, Y.: Unicorn: unified resource orchestration for multi-domain, geo-distributed data analytics. Futur. Gener. Comput. Syst. **93**, 188–197 (2019)
30. Zarrin, J., Aguiar, R. L., Barraca, J. P.. "Decentralized Resource Discovery and Management for Future Manycore Systems." arXiv preprint arXiv:1710.03649 (2017)
31. Khaneghah, E. M.: "PMamut: Runtime flexible resource management framework in scalable distributed system based on nature of request, demand and supply and federalism." U.S. Patent No. 9,613,312. 4 Apr. (2017)
32. Mirtaheri, S.L., et al.: Four-dimensional model for describing the status of peers in peer-to-peer distributed systems. Turk. J. Electr. Eng. Comput. Sci. **21**(6), 1646–1664 (2013)
33. Dueck, D.: Affinity Propagation: Clustering Data by Passing Messages. University of Toronto, Toronto (2009)
34. Gu, R., Becchi, M: "A comparative study of parallel programming frameworks for distributed GPU applications." Proceedings of the 16th ACM International Conference on Computing Frontiers. (2019)
35. Xu, W., Liu, P., Cheng, L., Zhou, Y., Xia, Q., Gong, Y., Liu, Y.: Multi-step wind speed prediction by combining a WRF simulation and an error correction strategy. Renew. Energy. **163**, 772–782 (2021)
36. Allen, G., Angulo, D., Foster, I., Lanfermann, G., Liu, C., Radke, T., Seidel, E., Shalf, J.: The Cactus worm: experiments with dynamic resource discovery and allocation in a grid environment. The Int J High Performance Computing Appl. **15**(4), 345–358 (2001)