J Supercomput (2014) 67:1–30
DOI 10.1007/s11227-013-0982-z

AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime

Ehsan Mousavi Khaneghah · Mohsen Sharifi

Published online: 16 July 2013

Abstract High Performance Cluster Computing Systems (HPCSs) represent the best performance because their configuration is customized regarding the features of the problem to be solved at design time. Therefore, if the problem has static nature and features, the best customized configuration can be done. New generations of scientific and industrial problems usually have dynamic nature and behavior. A drawback of this dynamicity is that the customized HPCSs face challenges at runtime, and consequently show the worse performance. The reason for this might be due to the fact that dynamic problems are not adapted to configuration of the HPCS. Hence, requests of the dynamic problem are not in the direction of the HPCS configuration. The main proposed solutions for this challenge are dynamic load balancing or using reconfigurable platforms.

In this paper, a vector algebra-based model for HPCS reconfiguration at runtime is presented and named AMRC. This model determines the element causing the dynamic behavior and analyzes the reason regarding both software and hardware at runtime. Some results of the presented model show that by defining a general state vector whose direction is toward reaching high performance computing and whose weight is based on the initial features and explicit requirements of the problem, as well as by defining a vector for each process in the problem at runtime, we can trace changes in the directions and uncover the reason for them.

Keywords High performance cluster computing · Reconfiguration · Dynamic problems · Vector algebra model

E. Mousavi Khaneghah (✉) · M. Sharifi
School of Computer Engineering, Iran University of Science and Technology, Narmak, Iran
e-mail: emousavi@iust.ac.ir

M. Sharifi
e-mail: msharifi@iust.ac.ir

1 Introduction

Traditional high performance cluster computing systems (HPCSs) are well known as solutions to reach the required performance of either complex programs or programs interacting with a large volume of data [1]. These systems use a customized initial configuration to reach the required performance. The configuration is a result of cooperation between experts of hardware, software, and the problem. They produce constraints, limitations, and features of the problem and design as well as develop a HPCS based on them to provide required computing power [2, 3].

A new generation of problems in the twenty-first century has dynamic nature and requirements [4–6]. The unknown nature of these problems causes dynamic behavior in processes at runtime. Dynamism of a process means that some of its requirements are unpredictable and formed at runtime as a result of the communication and the interaction with the environment of HPCS.

Based on Arthur Maccabe in [7], resource requests can be made in two ways: explicitly and implicitly. The former, explicit requests, are completely determined and stated in the program's source code. The latter, implicit requests, are generated at runtime and they are completely unpredictable. The programs with implicit requests are dynamic in nature.

On the other hand, we face dynamic computing platforms in the twenty-first century [8, 9] as well. Dynamic computing platforms mean platforms hosting different and heterogeneous resources leaving and joining a system. In these platforms, resource availability is changeable and these changes are unpredictable, thus cause dynamic processes.

Dynamism in programs and computing platforms violate the traditional model, that is, customized initial HPCS configuration. Approaches proposed for solving this problem can be categorized into two groups, the first of which aims to solve the dynamism in problems by building dynamic hardware platforms. Vast researches on building reconfigurable hardware platforms have been done. These researches aim to bring the same degree of flexibility represented in software into hardware. Many of these researches make use of FPGA [10, 11]. By the introduction of dynamic problems, efforts on adding runtime capabilities to these systems have been carried out. Among these researches, we can point to [12] whose authors introduced a partial run-time reconfiguration for computing systems. The challenge stated as the author's motive is related to the limitation of reconfigurable computing systems in satisfying big problems' needs when they request more resources than are available in the system. In such a situation, the underlying platform formed by reconfigurable hardware elements must be able to satisfy requests of the program at runtime.

The second group aims to solve the dynamism in computing platforms by building dynamic management systems [13]. Among these researches, we can point to the approach proposed in [14]. This approach aims to have problems based on homogeneous HPCSs, executive on heterogeneous and dynamic grid platforms by building an adaptive load balancing mechanism. In this research, Virtual Reactor Simulator, a well-known, important, and complex application simulating plasma chemical deposition vapor, is used as a test case and the Russian-Dutch grid as the executive platform. The proposed approach for adaptive load balancing in heterogeneous environments

works by minimizing inter-process communications and gaining the tradeoff between generating workload based on the capacity of processes. In this approach, two sets are defined: the first contains resource features and the second is the program's parameters. It is possible for these sets to be updated as soon as a change in resources or in program features occurs. For each obtained parameter, the corresponding load, based on available resources, is calculated.

In [15], a dynamic HPCS with dynamic state is considered as the underlying platform, and they investigate changes caused by sudden and asynchronous behaviors of communication, scheduling mechanisms, heterogeneity of processes, and workload changes, as well as effects of workload changes in performance of the load balancing algorithm. Based on these investigations, some approaches for improving workload algorithm in situations with high frequency of changes are introduced. They have proposed four rules, which must be considered in any load balancing algorithm. These rules are: load measurement, information rules, initiation rule, and load balancing operation. Following, they have investigated the effects of workload changes on each of these four rules.

The approach of this paper is based on transferring the concept of HPCS configuration from design time to runtime. In other words, the element responsible for HPCS configuration must be able to reconfigure the system at runtime as well. For this, we need to answer three questions. First, which is the element responsible for the runtime system reconfiguration based on newly created needs? Second, how can the elements detect the dynamic behavior (i.e., what is the criterion)? Third, which are the states causing dynamic nature? To answer to these questions, this paper proposes an algebraic model for reconfiguration of HPCSs at runtime (AMRC). To answer the first question, in the AMRC, the element responsible for HPCS configuration at runtime changes from an external entity (HPCS experts) to an internal entity (HPCS management system). In such a situation, since the goal of an HPCS expert is to reach the peak performance of HPCS, the HPCS management system must operate toward reaching the peak performance in executing the program by reconfiguring and managing units of HPCS. Reaching this goal (peak performance) in the best case means a process is able to use all the resources in the HPCS such as the local resources (Single System Image). In distributed computing, a Single System Image (SSI) HPCS is a cluster of machines that appears to be one single system [16]. In such a system, the processes use all of the global resources as local resources.

In order to answer the second and third questions, there must be a criterion based on which HPCS management system can detect a dynamic process as well as a measure on which this process is compared and its deviation from the HPCS configuration is calculated. The AMRC model represents a vector algebra-based math model [15] to determine the process whose direction is not in the direction of the HPCS configuration and which causes this mismatch. The represented model is based on the principle that the runtime HPCS reconfiguration is possible by matching the changed directions of the executive elements (i.e., processes) to the direction of the problem's general state vector (based on which the initial HPCS configuration is obtained).

This paper is organized as follows: Sect. 2 describes the preliminary concept of the model; Sect. 3 includes the related research in this area; Sect. 4 discusses customized configuration; Sect. 5 presents the responsible element of the runtime reconfiguration; Sect. 6 introduces the function of configuration at design time; Sect. 7 presents

the AMRC model for the HPCS reconfiguration at runtime; Sect. 8 discusses the proposed model; and finally, Sect. 9 concludes the paper.

2 Preliminary concepts

Regarding the fact that the AMRC in this paper presents a new level of abstraction for HPCSs to solve the reconfiguration challenges at runtime for twenty-first century programs, we will review some concepts of HPCS in this section.

2.1 High performance cluster computing systems

An HPCS contains of a set of loosely connected or tightly connected computers that cooperate together so that in many respects they can be viewed as a single system [17].

HPCSs have their own (just like other systems) causality and structure:

(A) Causality of an HPCS is that the system can execute a problem (which consists of some executive elements or processes) within the shortest possible time.
(B) The structure of an HPCS relates to what executive elements (processes) of the system? And, as viewed by the HPCS management system, on what basis are they defined? On the basis of what response structures do the executive elements interact with each other to reduce the total response time of the system?

2.2 The concept of customized configuration in HPCS

The configuration of an HPCS is identifying the response structures (at the design time or at the runtime) to requests so that an HPCS can achieve its causality.

In the twentieth century problems, since the problem expert demonstrates full knowledge of all aspects of a problem (nature of the problem; executive elements of the problem; and the characteristics, limits, and capabilities of each executive element), he/she designs an accurate response structure based on them. The duty of an HPCS management system is to utilize and manage the response structure that is determined by the HPCS expert during the life time of the HPCS.

In twenty-first century problems, there are three types of dynamic-interactive behaviors: dynamic behaviors in executive elements, a high complexity in communications among the executive elements inside the system or complex communication among the executive elements of system, and executive elements of environment. The expert does not have full knowledge of executive elements constituting the problem. In this case, the problem expert has implicit knowledge of the problem and, therefore, he/she cannot define an accurate response structure on the basis of characteristics, limitations, and capabilities of the executive elements. Therefore, the problem expert defines a responding structure to solve the problem on the basis of his/her own view about the problem generally. When an HPCS executes the program, the initial model of configuration offered by the problem expert incurs distortion because of the occurrence of one of three conditions that has caused the dynamic-interactive behavior.

2.3 Dynamic-interactive nature in 21st century scientific programs

In order to obtain a precise view on dynamic-interactive nature, we need to have a precise understanding of two concepts: the cause of dynamic nature and the time of the occurrence of dynamic behavior.

If we compare the twenty-first century programs' natures to the ones of the twentieth century that required the use of HPCS, we realize a very significant difference in the said program's nature. In scientific programs of the twentieth century, the rules dominating a natural phenomenon are exploited in a special field of science, and the aim of an HPCS expert in using the HPCS is to consider the program parameters against the high volume of data, or exploit the rules related to communication among parameters, or investigate the natural event in a shorter response time. The very important thing is the existence of rules on a natural phenomenon in a special field of science. Existence of such rules helps the system expert obtain full knowledge on the nature of the program that is being executed on HPCS as well as know what executive elements the program can be divided into, which helps the expert have more precision and dominance on advantages, limits, capabilities, and characteristics of executive elements. On the other hand, in HPCSs of the twentieth century, we observe batch executive behavior, which means the HPCS starts its activation against a specific program and only seldom changes occur in HPCS responding structures. It helps the expert obtain full knowledge about HPCS functions and also decide characteristics and limitations of the HPCS. In addition, all of this information (program nature and its executive elements and the HPCS) helps the expert precisely distribute the executive elements on HPCS and resource allocation to executive elements.

During the trend of executing the executive elements on HPCS, few distortions may occur on load distribution because of the executive program code nature executed on HPCS element. To solve this problem, the expert defines a fundamental activity called the load distribution on HPCS that preserves the load distribution model offered by the expert at the runtime using tools and means called migration, resource reallocation, monitoring, and synchronization.

That is, if we consider the set of activities of migration, resource reallocation, monitoring, and load redistribution as the duties of an HPCS management system, then the load redistribution activity on HPCS will be the axial activity based on which other activities are defined and executed. The load distribution on HPCS is executed on the basis of executive elements, and the HPCS management system defines the axis of activity based on executive elements existing in the system level that are the processes existing on HPCS level. The aim pursued by the expert in using the HPCS is to reduce the response time; therefore, the mechanism of load redistribution is designed in a way that whenever any one of the effective factors in machines eliminates the cause of HPCS utilization, the mechanism of load redistribution utilizes its own means (such as the process migration, resource reallocation, and monitoring) to preserve the main causality of HPCS utilization.

Meanwhile, twenty-first century programs try to exploit and determine the rules dominating over natural phenomena. The objective instance of such scientific programs can be observed as new weather forecasting or optic programs. In this condition, the expert designs the HPCS on the basis of general laws that are the basis

of developing the scientific program in this regard (such as the traditional rules of light and mass–energy) that is executing the program. On the other hand, lack of full and precise knowledge of the expert on problem nature, and consequently the requirements of executive elements causes the lack of a precise and specific view on the functional nature of HPCS. The practical factor that increases the complexity and causes lack of precise knowledge and dominance on the problem is the existence of unpredictable communication and interaction among the executive elements at the runtime. Existence of new interactions and communications in the system cause the initial responding structures to be unable to meet the scientific program needs and thus to be unable to execute the program.

Complicated interactions and communications are themselves the main cause of interactive-dynamic nature in twenty-first century programs and are the results of the following:

(A) During the program execution, the executive elements develop new executive elements (fork a process) allover system, and as viewed by the program expert, the newly developed elements either indicate the gaining of new effective parameters during exploitation of rules on natural phenomena or indicate a new communication between parameters of rules on natural phenomena.
(B) During the program execution, the executive elements (including the initial elements or the elements created during program execution) require new interactions and communications inside the system that are not considered in initial responding structures.
(C) During the program execution, the executive elements (including the initial elements or the elements created during program execution) require interaction and communication with the system-environment. This, as viewed by the program expert, means that the limits considered for exploiting the rules on natural phenomena are not correct or that there are parameters on the system-environment that are highly interdependent with the rules on natural phenomena (and consequently, the parameters that create and describe the phenomena). The communications and interactions of the system-environment cause scalability of the system (including the size, geographical, or managerial) as viewed by the HPCS management system.

Therefore, the problem question that we follow in this paper is what model HPCS management system can manage the HPCS to cope with distortion condition of dynamic-interactive behavior so that the HPCS causality is always established? How does the management model affect the fundamental activity of a traditional HPCS (load distribution)?

3 Related work

Two general policies can be used for reconfiguring the system to execute dynamic-interactive programs efficiently. The first policy uses the approach of Adam Smith's model of the invisible economic hand of the market [18] in self-regulating behavior of the marketplace. Based on a model derived and expanded from Adam Smith's model,

we can be sure that the HPCS certainly can achieve best configuration at another point in which the HPCS causality is established again. This other point is a new responding structure for HPCS in which problem execution elements interact and communicate with each other so the system again can run the program within the shortest possible time. In order to accelerate the process of achieving another balance point, it is possible to use numerous mechanisms such as load redistribution [19] based on dynamic nature or using the hardware that are capable of reconfiguration [20].

Some different solutions have been presented for load redistribution in different researches, and they have suggested the special purpose load redistribution model generally [21–23].

The first model emphasizes the fact that whatever caused the distortion creating factor affects the load distribution unit. Thus, it defines the mechanisms such as load redistribution capability at runtime on HPCS. Although the load redistribution mechanisms cause the HPCS to return to its causality establishment condition, the main and important challenge is dependency on a special program and exploitation of program distortion models. That means that the HPCS expert exploits the models that cause distortion by consecutively executing the scientific programs. He/she defines the mechanism in the load distribution unit so that the effect of said distortions on load distribution concept in HPCS is eliminated. Although this method initially seems to be an efficient and precise method, it cannot be generally utilized because of the following three challenges:

(1) Consecutive execution of program on HPCS to obtain distortions is somewhat expensive and time-consuming.
(2) The nature of identifying the rules on natural phenomena in exascale duration programs is moving toward identifying the unknown facts in science, which makes the identification of distortion models in this program complicated or in some cases impossible. Therefore, consecutive execution of this program cannot help us.
(3) One-dimensional attitude on load distribution. In this method, the main hypothesis is based on the fact that, notwithstanding the nature of distortion development in HPCS, the distortion necessarily and definitely will be effective on load distribution as the fundamental unit of HPCS; therefore, we have:

$$f(Disorder) : (Disorder_{space}) \rightarrow (Load\ Redistribution_{Space}) \tag{1}$$

In Eq. (1), which is used as the main equation of load redistribution model or dynamic distribution of load, the status effect called distortion (consisting of three states of A, B, and C) is only discussed on load distribution status. The said assumption may be deemed as an acceptable assumption, considering the simple modes of A and B states and ignoring state C; however, the existence of complicated states of A and the existence of state C cause the only impressible state from among distortion not to be limited to load distribution in HPCS level. The existence of state C causes attention to concepts such as models of communication between processes, as well as the capacity of the machines, benefiting from HPCS regarding scalability, and the existence or lack of existence of information about the whole HPCS as the impressible states.

The most important challenge of HPCS in using the approach of Adam Smith's model is due to the nature of HPCS; the HPCS is developed for executing one scientific program within the shortest response time, and the required time to achieve the new balance (new configuration) is not defined at the moment of designing and developing the initial configuration.

Indeed, the initial configuration is optimized because the system expert's knowledge of the system, the functional nature of HPCS, the cause of using the HPCS to execute the program (reducing the responding time), and the mechanisms of implementing the responding structures and the problem (separating the program to executive elements) results in the most optimized configuration possible in the system and the scientific program (problem). This concept, titled "Customized Configuration" in this paper, means the causes of optimization of initial configuration. Considering the concept of customized configuration with respect to the fact that time is the main cause of causality in HPCS, we can obtain the clear result that HPCS cannot use Adam Smith's model because this process is time consuming.

The second policy that we used in this paper is based on determining the dynamic-interactive behavior and reconfiguring the system based on customized initial configuration at the distortion event. With this policy, we need to use a general mechanism for identification of distortion agent and confrontation with distortion nature.

The limitations mentioned above for the equation of load and the complexity in scientific programs in exascale Computing [24] demand that we be needful of a solution that: (1) is able to exploit the cause of distortion in every scientific program without using the initial assumptions; and (2) is able to investigate the effects of causes of distortion in HPCS in various perspectives, and on the other hand, can offer an applicable model (without using a general mechanism distinctive from scientific program nature and on executive elements) for reconfiguration of HPCS when confronting with distortion caused by dynamic-interaction nature. Reaching this solution requires:

(A) Exploitation of the initial customized configuration presented by the system expert.
(B) Definition of the HPCS based on beneficiary elements in HPCS configuration.
(C) In today's HPCSs, the definition of the HPCS management system about the executive element is equivalent with the definition obtained from the role of the executive element in HPCS based on load distribution fundamental activities. Meanwhile, in order to contend with distortion of scientific programs of the twenty-first century, a definition by the HPCS management system should exist that is independent from the scientific program, is distinctive from the rule of executive element in load distribution, and is developed based on the beneficiary elements in the HPCS configuration. The HPCS management system always considers the executive element (or process) as active on a machine based on load distribution indexes with the fastest response time possible. In case we accept this assumption that the initial customized configuration of HPCS is the best configuration implemented from scientific program on HPCS, the executive element should describe its condition regarding the initial configuration during the specific time, and it should contain a mechanism to determine the deviation from or compliance with the initial configuration.

(D) Defining a general management mechanism without dependency on time and place but dependent only on the new definition of HPCS, and defining the executive element condition so that it can change the HPCS elements or the executive condition of executive element to return the executive element and the initial customized configuration when there are faults in executive element condition conformity and function.

In other words, the said general management mechanism is a mechanism which only:

(1) Depends on HPCS (based on agents effective on configuration) and the definition of executive elements against the configuration.
(2) Is able to return to its initial balanced condition whenever inconformity of one or several processes occurs due to distortion or changes in effective elements in configuration or executive nature of executive elements.
(3) Is the run-time mechanism and occurs in HPCS management system level (especially by the executive elements structures).
(4) Is ineffective on execution of traditional programs, and maximum compatibility can be obtained by the traditional HPCSs.

Thus, the discussion challenge is how to define the HPCS on the basis of elements effective on configuration so as to be capable of defining the executive elements based on configuration status so that the executive element can describe its own condition with respect to conformity with or deviation from initial customized configuration. This way, when distortions occur, the HPCS management system can change the configuration status or the executive condition of the executive element to make the executive element conform to the initial configuration at the runtime.

In this paper, the said mechanism and the executive element definition is based on vector algebra [25]. The most important characteristic of vector algebra is that it can be used regardless of time and place. When the element (the executive elements of the entire operation) is defined based on vector algebra, the description is conducted without considering the time and place. The most important ability of vector algebra in describing the element is based on two concepts of weight and direction of the element. On the other hand, the vector algebra can be defined on the basis of two general operands, each one of which can conduct activities on the described elements based on vector algebra, which results in a new condition.

4 Customized configuration model

Customization in HPCS means designing and developing an HPCS based on the requirement and features of the problem [26]. The pattern of designing a customized HPCS is as follows: the problem's experts (problem creator and system programmers) specify the problem's constraints, features, requirements, and execution model. Then, based on these, the HPCS experts design and develop a customized HPCS to meet the conditions and the features of the problem [27]. A simple model for the customization can be as follows, and we named it "five-phases design model":

Phase 1: Problem and HPCS experts cooperate.

Phase 2: Problem's experts form a set named Problem Technical Accounting (PTA), based on problem features, including: its constraints, special features, requirements, and technical information.
Phase 3: HPCS experts design and develop the HPCS management system based on the PTA set and set of available resources in the system. The HPCS management system must control system resources in a way that each process, within its lifetime, behaves with the entirety of resources in the system as if they are local resources.
Phase 4: The nature of the problem nature is static, and its processes are static as well. The HPCS manager executes processes based on the PTA set, which is static.
Phase 5: HPCS experts leave the system, and the system enters execution phase.

Therefore, the concept of an HPCS configuration can be defined as the set of operations performed by the system experts. These operations try to configure resources of the HPCS in a way that the PTA set requirements for achieving high performance are met. The HPCS management system is to control processes' executions and to provide them the required resources.

In this system, three elements play the main roles: the HPCS experts, the PTA set, and the resources set. A function for the design time system configuration (i.e., the configuration function from the HPCS expert's point of view) can be defined by considering the problem as the function's domain and its solution as the function range (problem-solution). The execution of this function matches the resources set to the PTA set.

This function must have the following features:

1. A mathematical function whose design and definition are not considered as constraints for the HPCS.
2. The function execution must be handled by an existing element within the HPCS. In other words, a new element, whose addition to the system increases its complexity, must not be responsible for the function's execution.
3. This function must be adapted to the dynamic nature of the PTA set and variable resources with the goal of reaching high performance (this means in the lifetime of the HPCS, its final goal must not violate).

5 The responsibility of the HPCS reconfiguration at runtime

The traditional model was well qualified in achieving high performance because both the resources set and the PTA set were static. By using these two static sets, HPCS experts were able to drive out the match model based on how resources could be configured in a way to achieve high performance.

By the move of the PTA set toward being dynamic, we have to change the fundamental element responsible for HPCS reconfiguration. Having another look at the assumed scenario for the HPCS configuration at design time, it is worth noting that one of the basic reasons for being unable to handle the dynamicity of the PTA set in the traditional configuration model is initiated at Phase 5, by removing the element responsible for configuration of the HPCS (i.e., the HPCS experts). Nevertheless, even assuming that this element is not removed from Phase 5, since this element is considered as an external entity to the HPCS, the following scenario with an added phase 6 is defined.

Phase 6 The HPCS expert is notified that the nature of the PTA set has changed. On the other hand, the HPCS is executing processes. At this moment, either the HPCS expert must change its position from an external entity to an internal entity to reconfigure the system or direct the system to a suspend state, after which it can reconfigure the HPCS based on the new PTA set, which is of course very time consuming.

Indeed, if we change the limits of HPCS and consider the system expert (which undertakes configuring the HPCS) as an internal element of the system, then this element should reconfigure the HPCS in case of occurrence of distortions A, B, or C in the system, on the basis of a new PTA. This is essential to note that Phase 6 is an abstract phase, and it is impossible to define such a phase in the real world. What we look for in Phase 6 is the element capable of reconfiguring the system in runtime.

Therefore, it is not beneficial for the system to be reconfigured by the system experts. However, the sixth phase points out an important matter: the changing of system experts from an external entity to an internal entity. As a result, the system manager, as an internal entity, not only controls execution of the operations (which consequently requires process management) but also takes the responsibility for the system reconfiguration and plays the role of the HPCS experts within the runtime.

6 HPCS configuration function at design time

We first investigate the HPCS configuration from the view point of the HPCS manager as the deputy of the system experts. Assuming the responsibility of the HPCS configuration in the design phase is with the system manager, the concept of configuration is simplified to the introduction of a single system image for a process. That is because the HPCS management system handles processes in order to configure the system to achieve the required performance. The HPCS management system expresses the concept of system configuration in the form of the following function:

$$\text{“}\textit{Configuration Function}\text{”} : (\textit{Set of resources}) \xrightarrow{\text{Mapping}} (\textit{SSI}) \tag{2}$$

The above function expresses that the elements of the resources set must be managed in such a way that at the end, the process (processes) being executed on the HPCS faces a Single System Image (SSI), which means the process uses all the resources in the system including local resources. By such a management over the resources set, the system configuration can be customized toward achieving high performance. Therefore, the problem-solution concept can be defined by two more perceptible concepts in HPCSs. These two concepts operate under the control of the HPCS management system. The first concept is the resource set of the HPCS as the problem and the second concept is SSI as the solution. This is assumed in Eq. (2) that the function domain (Total resources existing in HPCS) plays the role of a vector space where the members equal with resources existing in HPCS, and their weight demonstrate the number of existing resources, and their direction is positive when used in HPCS and negative when not used. The range of this function is achieving the aim of SSI. In the following paragraphs, we will note that with respect to the fact that implementation of the AMRC model should be on kernel level of operating system and as viewed

by the operating system the resources existing in system level should be classified in four general groups of input/output, memory, processor, and inter-process communication [28, 29], and the file resources (or storage). The HPCS should be able to define quadric vector sub-spaces for each process so that the process deems the elements of each one of the quadric vector sub-spaces as its own local resource. Indeed the HPCS should develop the SSI vector space in quadric sub-spaces in each one of which there are the resources of the sub-spaces and its weight shows the utilization amount and its vector is positive in case it can provide the SSI concept.

This function is considered as an initial function based on which a second function for reconfiguring the system at runtime is defined. The mentioned function has some ambiguities. For example, this function operates as a mapping function; however, although mapping can be realized at design phase, how it can be realized at runtime remains an open issue. Another example is the definition of SSI and whether SSI can make it reach customization. The final example concerns the resources set and their status at runtime. We try to disambiguate these as follows.

Equation (2) represents the main goal of the HPCS management system. The main duty and the final concern of the HPCS management system is to execute a function whose domain covers a space, called available resources in the system, and range covers a space, called the HPCS goal.

Equation (2) shows the model of configuration that HPCS experts use for the system to achieve high performance to execute the scientific programs. This model is used in of twentieth century scientific programs because of complete competency of the expert on the programs and also the HPCS at the time of designing the HPCS. In scientific programs of the twenty-first century, the HPCS expert cannot use Eq. (2) precisely and completely at the time of designing the HPCS because of the dynamic-interactive nature (distortion models) and because the expert is not knowledgeable in programs. Obviously, transferring Eq. (2) to the runtime is so complicated and difficult that in some cases it results in total failure in causality of HPCS. However, as mentioned before, the customized configuration model of HPCS is the best model used for configuring the HPCS; therefore, in the following paragraphs we will try to exploit a model after precisely defining the range of Eq. (2) to describe the HPCS and processes as the executive elements so that it can be used as the criteria function at the runtime to protect the customized configuration of HPCS.

At this point, we would like to describe the situation of this function regarding 'n' member nodes in the system. A HPCS can be usually made up by n sub-systems. These n sub-systems can be either complex or simple systems subsequently. Simple sub-systems involve a single independent computer, and complex sub-systems involve any type of systems formed to provide high performance. From the standpoint of the HPCS management system, with responsibility of reconfiguration of the HPCS at runtime, the whole system can be considered as 'n' sub-systems. Based on the situation of all the elements within the HPCS, the HPCS management system can perform the configuration operation. Therefore, in the AMRC model, from the HPCS management system point of view, the whole system can be considered an n-dimensional space. In such a situation, we have:

$$(Causality\ of\ HPCS\ Manager) : (Set\ of\ Resources) \xrightarrow{n\text{Mapping}} SSI \tag{3}$$

We need to investigate the final function of the HPCS management system more accurately. This function is responsible for the HPCS management system and works toward solving the problem, meaning HPCS configuration. In the following, its constituting elements are explained in detail.

7 The AMRC model for HPCS reconfiguration at runtime

In this section, the detail of the AMRC model is described and the principles, basic concepts, and operators of the model are discussed. First, we will define three concepts needed for configuring the HPCSs at the runtime. As mentioned before, the HPCS management system is able to handle the reconfiguration process at the runtime on behalf of the system expert (Phase 6), and on the other hand, Eq. (3) expresses the ultimate aim of the HPCS management system. Thus, we need to investigate Eq. (3) further on the basis of fundamental concepts so that we can define the vector generating space of HPCS, and consequently, the process (executive element) as a sub-space in the vector generating space of HPCS; and finally, specify and determine the vector algebra that is the HPCS management system based on two concepts of system space-process for reconfiguring the HPCS at the runtime. The model of AMRC is indeed the developed form of Eq. (3) based on vector algebra and provides the possibility for the HPCS management system (that is based on it) so that at the runtime, in addition to performing the traditional tasks, it can identify the distortion models and manage them on the basis of returning back to the customized configuration.

7.1 Principles

As described by Eq. (3), the model explaining the HPCS management system performs a mapping between two spaces. To achieve the high performance, the HPCS management system must be able to match the elements of the set of resources to either the problem features or PTA set. This means that at any instance within the lifetime of the system, the configuration model will never contain an execution pattern of processes in which system's executing elements do not match with the features of the PTA set.

The first principle, *named T1*, is derived from the underlying philosophy of the HPCS experts that consider HPCS as a problem solver that a scientific problem is as an input, and outputting is a solution to it. Therefore, our first principle is to map the resource requirements of a problem space to a solution space by adapting the capabilities of the solving nature. The adaptation is done by the processes that run computer programs to solve the problem. T1 comes from a systematic approach [30] to HPCS philosophy.

T1 principle is the most important principle in describing the challenge of twenty-first century dynamic problems. Therefore, in this paper, the main problem is the violation of T1 at runtime of the above processes and lack of a model for the runtime HPCS configuration.

T1 principle investigates another important point as well. The mapping model in the descriptor function of the HPCS, and furthermore, in the subsequent sections of

the paper, must cover the customization concept in the two sets: the set of resources and the PTA set. However, a model which can transfer T1 principle from design time to runtime can be considered a reasonable model for HPCS configuration in interacting with dynamic problems.

We know that any solution to a scientific problem requires computer processing, memory, I/O, and/or stable storage (e.g., disk) resources [28]. To allow for dynamic configuration of our chosen solver, the HPCS, we use this feature as our second principle. We partition an HPCS into four parts, each part responsible for one of the types of resources. Note that each HPCS can be constructed from any number of HPCSs. This is the nice property of a single system image supported by the system.

HPCSs provide a single system image to users and programs. Using the vocabulary of system science, the implication of single system image is that every element (member) of a system plays a particular role and performs a particular set of operations (determining its causality) in orchestration with other elements to accomplish the main goals and causalities of the HPCS, namely, solving a given problem. The elements are thus no more general purpose when they are assembled and aggregated as a system to solve a certain problem, although they provide a single system image to outsiders; they are special-tailored to the requirements of the problem.

The second principle, *named T2*, investigates the two basic and important points in the function introduced in Eq. (3). The first point is that the dynamic nature of twenty-first century processes is not mutually exclusive to their processing behavior (dimension). Any change in the four-dimensional process management expressed in T2 principle can cause violation of customization and T1 principles. The second point is that SSI concept can only be considered as the goal of an HPCS if it can be applied to other dimensions of a process.

The third principle, *named T3*, gives a general categorization of the elements playing roles in the HPCS. According to this principle, the effective elements in the HPCS and heterogeneity of elements can be classified into four general spaces. Three of them are hardware, application software, and system software, and they are commonly mentioned in some researches in the HPC area [31–33]. The fourth space that was considered in design of exascale computing is Functionality [4].

Looking at the road map of exascale computing, we can realize that the scientific programs in the new generation of exascale computing systems are of dynamic-interactive nature, and thus, we will consider the concept of Functionality in addition to the above three concepts.

In fact, HPCSs are built from independent communicative computers with the same or different hardware features and capabilities (H). They may run under the same or different system software (SS); they may execute the same or different (parts or whole of) application software (AS), and they may provide the same or different functionalities (F). This is to say that the elements (members) of HPCS can be heterogeneous in H, SS, AS, and F; this is our meaning of heterogeneity.

7.2 General concerns of the AMRC model

Based on the systematic investigation, two systems are considered equal if first, the generator space of the two spaces are equal, and second, the operators of the two

systems that describe their main activities (functions) are equal. The AMRC model uses these basic principles to define AMRC vector algebra. Furthermore, The AMRC model uses the vector algebra as a tool for determining the element, which is not in the same direction of the current configuration of the HPCS. The AMRC model can describe the reason of changing the directions of the vectors at runtime by using linear algebra.

The general approach in the AMRC model is that the HPCS management system must be able to determine new changes and requirements of dynamic processes. The HPCS management system must also compare these changes and requirements to a main indicator existed in the time of system's stability.

T1 principle is one of the main solutions for this problem. This principle, based on mapping concept, at any instance of time investigates whether a change in the domain of the function causes reaching its range or not. T2 principle states: According to the four groups of resources, determine in which direction domain changes occur, so that the HPCS management system can control these changes toward reaching the range. Now this question arises: What is the criterion that the HPCS management system can use to make measurements? T3 basic principle is applicable solutions to this problem.

The general model of the AMRC is based on vector algebra. Vector algebra has some interesting features including that it can be applied both in scalar and vector areas. The following elements exist in the AMRC model:

1. The AMRC model produces its generator space based on Eq. (2); this generator form of function is equal to a generator form of vector. However, there is a principle stating that if two different systems use the same generator form, then these two systems are equal. As a result, general rules of the two systems are equal, and consequently their operators can be equal, as well. Using this principle and the AMRC generator form of vector, the AMRC model can define the operators of the AMRC vector algebra. Again, using the mentioned principle, the AMRC model can state that the above operators are held (or are operational) in the space in which its generator form equals the space of the generator form of Eq. (2). Therefore, a set of operators is obtained that can be used for finding the state of the vector of each process and also specifying deviation of its direction of the general state vector.

2. In the AMRC model, a vector, called general state vector, is defined for the whole system. This vector direction is based on the HPCS management system, which must control involved elements of the system so that the required performance of the program is achieved. This vector can be figured out based on different approaches including: the program's initial information or the system's initial configuration information (information provided by the experts based on which system was configured), explicit information provided in the program. In this paper, it is assumed that this vector can be obtained based on the information explicitly expressed in the program about the required resources.

3. The AMRC model adds to the data structure of each process created in the system some fields called Process Vector fields. These fields can be controlled by the HPCS management system. These vectors describe each process's state regarding effective elements in the HPCS.

4. The AMRC model defines some operators for these vectors. Based on these operators, the HPCS management system can produce some information about the

process's state that has caused the system to be in a disordering situation and also about those elements of the resources set disordering as a result of interoperating with that process. By using the T3 basic principle and by making some changes in the state vector of the process and its interoperations with the HPC effective elements, stability can dominate the system again.

7.3 General state vector

One of the most important elements in the AMRC model is the general state vector, or vector I_k. Vector I_k represents the state of the problem to be solved by the HPCS against the elements of the set of the system's four groups of resources (described in T2 basic principle). Vector I_k in the most general form can be represented as in Eq. (4):

$$I_k = I_{k_{IO}} + I_{k_F} + I_{k_P} + I_{k_M} \quad \text{As '+' is vector plus} \tag{4}$$

A program generally has four types of needs to resources from view of the system's manager (Operating System or any HPCS management systems). Therefore, the requirements of programs can be investigated in quadric vector spaces, and the total vector amount of these four shows the total requirements of the program to the resources existing in the system. Equation (4) shows segregation of program requirements to quadric vector sub-spaces. The weight of these vectors shows the significance level of each one of these main classes, and their direction is always positive (because they should essentially respond).

As shown by Eq. (4), general state vector contains four vectors. The AMRC model assumes that vector I_k is actually the sum of the four I_k vectors in memory, process, input/output software, and file spaces. It should be noticed that interpreting vector I_k in a specific space will show (generally and without considering details) the expectations of a problem from the set of available resources in the system for reaching its required performance. In this paper, to obtain I_{k_i} vectors, explicit requirements of programs to resources are used.

In the AMRC model, required information for constituting of I_{k_i} vectors are stored in data structures at initial customized configuration of the HPCS performed by experts and using available information provided by the programs.

7.4 Generator space of the HPCS management system

According to T1 principle, mapping means matching the resources set to the PTA set. Having the general activity vector, a more accurate view of the PTA set can be obtained. As a matter of fact, the PTA set is the generator space of general state vectors. Each general state vector is an interpretation of the PTA set regarding one of the four dimensions stated in T2 basic principle. We can use T3 basic principle and heterogeneity in the four dimensions to define any machine (or any sub-system in the HPCS). So, we have:

$$\forall \textit{Machine} \in \textit{HPCS set}: \textit{Machine Space in HPCS} = (H, SS, AS, F) \tag{5}$$

Equation (5) gives a description of elements within a machine playing a role in the execution of the problem (program) to be solved (executed) by the HPCS. Now, by

substituting the original domain of the function presented in Eq. (3) with T1 basic principle, we reach Eq. (6), which represents the state of each HPCS in the mapping process.

Substituting Eq. (3) in Eqs. (2), (4) presents a more meaningful definition of our theory, namely to map a network of independent machines into an HPCS to solve a given problem. Space 'C' represents the final goal of an HPCS. As shown in Eq. (2), from the standpoint of HPCS experts, space C is equal to the problem solution.

$$Function(Causality\ of\ HPCS\ Manager) : (H, SS, AS, F)^n \xrightarrow{\text{Mapping}} C \tag{6}$$

Equation (6) is derived from T3 principle and represents the original domain in T1 principle and those elements of a machine involved with HPCS management system. In order for the system to be able to solve a problem with the high performance (required), these elements must be configured by the HPCS management system. Although this function gives a description of the basic activity of the HPCS management system, in practice it covers the communications between the effective elements of the HPCS and the PTA set.

As stated previously, if we look at the range of a function from the standpoint of HPCS experts, the range must be interpreted as a quality and thus will not be measurable. On the other hand, the AMRC model needs the range of a function to be a quantity so that using its measurable nature (like the quantity used in Eq. (6) to represent a function's domain), the generator form of the HPCS goal can be created. As stated by Eq. (3), from the standpoint of the HPCS management system, the range of a function is equal to SSI. However, as T1 principle, the HPCS from the view point of the executive elements must match the two sets, set of resources, and the PTA set. On the other hand, according to T2 principle, SSI concept must be applied in four dimensions. That is because customization concept or an equivalently matching set of resources to the PTA set must be performed in four dimensions. In addition, it must be noticed that T3 principle determined the set of resources and T1 principle determined the elements of the PTA set. Generally, categorization of the requirements of a problem is provided by T2 principle, and the PTA set contains four sub-sets, each representing requirements of the program to execute. Nevertheless, in the real world, each line of a computer program executes an activity related to one of the four sub-sets expressed in T2 principle. Thus, based on T2 principle, we expect that SSI concept contains four sub-spaces: $SSI_{I/O}$, SSI_{Memory}, SSI_{File}, and $SSI_{Process}$.

$$\begin{aligned} &\text{Causality of } HPCS \text{ Manager} : (H, SS, AS, F)^n \\ &\rightarrow (SSI_{I/O}, SSI_{Memory} SSI_{File}, SSI_{Process}). \end{aligned} \tag{7}$$

One of the most significant and special points is implementing the HPCS configuration to Eq. (7). The range of Eq. (7) is indeed the developed mode of range of Eq. (2). In this mode, considering the hypothesis T1 we can convert Eq. (2) in the range to Eq. (7). Depending on how the scientific program nature should be and which one of the four aspects of resources is needed for SSI, the HPCS expert can decide on selecting or not selecting the quadric sub-spaces of the range of Eq. (7). The total weight (the weight allocated by each one of the quadric spaces to exploit the

total weight of SSI) and also defining the vector direction of each one of the quadric sub-spaces of SSI space are the two main means of the HPCS expert for configuring the HPCS. In the programs of the twentieth century, the HPCS expert can provide an efficient map from HPCS space as the ultimate space of HPCS that is SSI, considering his knowledge of the programs and HPCS nature at the time of system designing with the range changes of Eq. (7). In scientific programs of the twenty-first century, the AMRC model can decide for processes execution changes by investigating the vectors of processes and system mode vector (the vector of estimation of the quadric sub-space of SSI). For example, in a general observation and without considering the details, it is possible to realize that in the case of distortions causing mode C, the weight, and direction of vectors existing in $SSI_{Process}$ and $SSI_{I/O}$ sub-spaces will have more changes.

Equation (7) expresses the final goal of the HPCS management system according to T2, T1, and T3 basic principles. As shown by Eq. (6), the final goal of the HPCS management system is to map the space $(H, SS, AS, F)^n$ to the $(SSI_{I/O}, SSI_{Memory}, SSI_{File}, SSI_{Process})$. According to the T1 principle, the function is a map function, in which the initial mapping of the sets of resources to the PTA set is expressed in the form of the above mapping. The range of this function is derived from the T3 principle and its domain from the T2 principle. At this moment, we need to produce the generator form of Eq. (7) by using spaces $(H, SS, AS, F)^n$ and $(SSI_{I/O}, SSI_{Memory}, SSI_{File}, SSI_{Process})$. To describe the final goal of an HPCS from the perspective of its management system, it is worth noting that this final goal is defined based on these two spaces.

$$HPCS \text{ Manager} = (Effective\text{-}Elements, C) \tag{8}$$

Equation (8) states firstly that the HPCS management system is considered a subsystem in the HPCS. Secondly, it states that an HPCS is designed based on the spaces 'C' and Effective-Elements. In fact, space Effective-Elements is the domain of function in Eq. (7).

Equation (8) discusses a form based on which the configuration concept of an HPCS from the view point of its management system is created. In addition, it discusses the AMRC algebraic vector and introduces an algebraic vector model to determine the element not in the same direction as the configuration of the HPCS and gives the factor of this matter at runtime.

In the previous section, it was stated that the generator form of the final goal of an HPCS from the perspective of its management system, which is equal to the configuration model of the HPCS from that element's perspective, is equal to Eq. (8) as well. The AMRC algebraic vector model investigates situations in which a process whose direction is not the same as that of the initial configuration of the HPCS is formed and tries to find the factor of this matter.

In the AMRC model, each one of the twelve vectors descriptor of problem has direction and weight according to customized configuration of the HPCS, and it is changed during the HPCS life and based on program instructions execution, and also it is able to investigate the changes and return back to primary balance mode.

The HPCS management system, regarding each instruction of the program executed by executive elements (or processes), investigates the resulted changes and

Pseudo-code 1 General scheme of the AMRC functional model

```
Q, Initial 12 vector based on customized configuration (HPCS-2Phase)
        For each executed instruction in each machine
        {
        If direction of the vector is changed {
                        Define disorder_state;
                        Use operator Δ to solve;
                }
        If value of vector is changed {
                        Define value_disorder_state;
                        Use operator # to solve;
        }
Q1.Update information ();
```

first determines whether change caused any distortion mode in the system; second, in a more general approach, tries to return the changed direction of vector (caused by the instruction execution) to the initial direction (the direction according to the customized configuration).

The Pseudo-code 1 is the operation that is executed by the HPCS management system based on the AMRC model. This Pseudo-code contains two significant points: first, it should be able to describe the initial mode of the twelve vectors problem descriptor on the basis of a set of information by using Eq. (9) and Eq. (10) and a concept called the two stage model of data exploitation (PTA) in the AMRC model; second, in the AMRC model, any kind of change caused by program instruction execution is known as a distortion. One of the most similar activities to the AMRC model are the models that exploit the distortion triple distortion modes in HPCS and define a set of approaches based on executing the program several times to encounter the triple distortion modes imaginable in the HPCS management system performing twenty-first century programs. Although this method precisely describes the nature of triple distortion modes in twenty-first century HPCS, its most significant challenge is increasing the response time. In order to solve this problem in the AMRC model, the first supposition of the HPCS management system is based on the fact that execution of each instruction by each one of the executive elements creates a distortion mode in HPCS. Depending on whether the instruction execution leads to changing the direction or weight of the vector, two operation of Δ or # are defined in the AMRC model based on vector algebra. In case instruction execution creates a new vector that does not align with the primary vector or its weight is not in conformity with the primary vector, the management system changes the elements of HPCS to correct the mode and return them to the initial customized mode.

The HPCS management system is defined in each machine that is a member of HPCS, and in the Pseudo-code 1, only the Q1 and Q2 lines are executed generally and by the machine start the operation and the rest of the program codes are locally executed inside each machine.

In the following, we will discuss more precisely the concepts of the HPCS-2Phase model and Δ and # operators.

7.5 HPCS-2phase model

If we suppose $Problem_v$ is a sub-problem resulted by Δ and # operators executed on two vectors defined in HPCS, then the HPCS management system uses the problem description concept based on HPCS generator space (in order to describe the problem and also determine the operational nature that should be executed on HPCS) against the management of program execution trend (based on the customized configuration and also the fundamental concept of load distribution). Determining the sub-problems based on vector algebra requires converting the main problem to some vectors (with specific weight and direction). In the AMRC model, we need to map the problem from its traditional definition to a definition based on vectors descriptive space.

Equation (9) shows that in the AMRC model the HPCS management system (based on the model defined in Eq. (10) maps the $Problem_v$ from the traditional definition space of the problem (the program that should be executed by the HPCS management system) to the vector descriptive space:

$$F(manager) : (Problem)_v \xrightarrow{\text{Mapping}} (Problem_{Vector})_v \tag{9}$$

Equation (10) shows a model that is used by HPCS management systems to analyze the problem into the sub-problems executable in HPCS based on beneficiary machines. Each beneficiary machine in HPCS are described and determined on the basis of four generating spaces. $Problem_v$ can be partitioned into twelve sub-space problems. Equation (10) gives these twelve sub-spaces. As you can see:

1. Any problem (program) needs hardware in order to be solved (executed). Assume that all hardware devices can be categorized into four groups: input/output devices, memory, processor, and storage (or file). Let us show them by H_{IO}, H_M, H_P, and H_F, respectively.
2. Any problem also needs system and application software in order to be solved. System and application software are managing or performing some activities related to the four groups. Let us show them by A_{IO}, A_M, A_P, and A_F; and S_{IO}, S_M, S_P, and S_F.
3. 'n' means number of elements in Effective-Elements set.

The AMRC considers any activity 'k' (a sub-activity of process 'v') occurring within a machine to be a descending or ascending vector for each set. This descending or ascending vector is created because of the occurrence of activity 'k', not only causes the current status of one of the problem v's sub-problems to be changed in context, but also in its movement toward completion of the sub-problem and in reaching the final state.

$$\begin{aligned} Problem_v = & \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{H_{IO}} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{Hp} \\ & \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{H_M} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{H_F} \end{aligned}$$

$$\cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{A_F} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{A_{IO}}$$
$$\cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{A_M} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{A_F}$$
$$\cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{S_F} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{S_{IO}}$$
$$\cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{S_M} \cup \bigcup_{i=1}^{n(Effective\text{-}Elements)} Problem_{S_P} \quad (10)$$

The problem requirements can be classified in four dimensions so the image of each dimension on each generating spaces of the HPCS should be obtained. Equation (10) is the image of each one of the quadric spaces of activity requirement in each one of the triple real generating spaces of HPCS. Therefore, if we want to describe a problem (set of requirements that should be responded to by the HPCS) based on the AMRC model, the problem will be concerted in duodecimal vectors in three spaces. The significant point is the weight of the vectors and also their direction. The weight of each vector is the significance of vector in solving (executing) the problem (program), and its direction is always positive.

In the AMRC model, Eq. (10) is used to calculate and define two general groups of vectors. The global operation starter machine uses Eq. (10) to calculate the general vector of the operation, and the HPCS management system in each machine uses Eq. (10) to calculate the vectors that describe the beneficiary element role in global operation. So that each unit of HPCS management system in each member machine of the system uses Eq. (10) locally to calculate the vectors describing the sub-problem(s) that is being execute inside the machine.

In the AMRC model, the HPCS-2phase model is used to calculate the weight of duodecimal vectors of problem description (or sub-problem) by the management system. Both the management system of the machine that starts the global operation and the beneficiary machines of HPCS at the time of formation of the problem describing duodecimal vectors (or sub-problem) consider the vector's direction positive. The HPCS-2phase model is based on the fact that it is feasible to obtain the data related to vector weight from either two general groups of user data (PTA) or the data provided from initial data structures.

Based on this model, a program in its most general status contains two sections of the initialization structure (the assignment of an initial value for a variable) and the computation structures. In twentieth century programs, these two sections are completely separated from each other, while in twenty-fist century programs, the initialization structures may appear again during the program execution trend due to dynamic-interactive nature.

Classification of scientific programs in two sections of initialization structures and computation structures helps us:

(A) Obtain an implicit view on weight, computation, communication, and also interactions on the basis of primary initialized structures of the program.

The primary initialized structures demonstrate the initial data amount that the problem expert has regarding the problem nature and functionality of HPCS. The data existing in initialized structures offers a suitable view (explicit or implicit) of the problem nature and also regarding the HPCS to the management system in the AMRC model.

In the HPCS management system based on the AMRC model, the initialization structures are used as elements to determine the vector direction. In order to decide on direction of each duodecimal problem describing vectors, the structures of initialization are used. The information existing in initialization structures is used as a part of information utilized by HPCS management system (including the management system in starting the machine or in the HPCS beneficiary machines) to determine the direction of duodecimal vectors. The amount of information of initialization structures in the scientific program constitute a small part of the scientific program (compared to computation structures); however, in the HPCS management system based on the AMRC model that should be implemented in kernel level of the operating system, data structure of kernel change is due to initialization structures and are able to offer an acceptable view on the weight of duodecimal vectors. Consequently, this is necessary to keep in mind that when a simple initialization on the level of program is transferred to the kernel of the operating system, it activates structures of various data such as memory, process, and even in some cases the I/O data structure.

The amount of exploitable data in a simple initialization on scientific program level in kernel level is as much as the management system (the starting machine or the beneficiary element) can decide upon the weight of vector. On the other hand, in the AMRC model, using the primary information (user-initialized information in the machine that starts the global operation) and also the dynamic nature of the weight of the problem (sub-problem) describing duodecimal vectors help the weight of vectors move toward the real weight during the HPCS life.

(B) Computation structures involve some information about the functional nature of the scientific program.

The information existing in computation structures in the AMRC model is used for calculating the changes resulting from execution. Execution of each instruction by a process inside the local machine is as a change in duodecimal sub-problem describing vectors level. The management system of the local machine starts evoking the #, Δ right after execution of an instruction by the local processor in order to realize whether the change made in weight or direction of problem describing vectors. The significant point is that the local management system decides on which duodecimal problem describing vectors should the changes made by instruction by process according to what data structure has the process changed on operating system kernel level. This is possible because of the implementation of the local management system on operating system kernel level.

7.6 Operator # in the AMRC model

What is our set of vectors in the vector space? The operation of a process running on an HPCS machine in the *Effective-Elements* set is considered a directional vector (we

call it *Saurus*). Operations by all running processes make up our vector space. We define two directions for the vectors—positive and negative. The positive direction of a vector shows that the requirements of the process denoted by this vector are in the same direction of the vector representing the capability of the HPCS in its customized configuration. The negative direction of a vector shows that the requirements of the process denoted by this vector are not in the same direction of the vector representing the capability of the HPCS in its customized configuration.

Based on defined sets of scalar values and vectors, we can now define two vector operators. Equation (11) defines the operator '#' for every two members of "Saurus".

$$\forall i,\ j\ \in\ Saurus : v = i\#j\ \therefore v \in Saurus \text{ and } v \text{ presents cooperation between } i, j \tag{11}$$

Equation (11) shows that the two members of Saurus such as 'i' and 'j', representing vectors of two processes in the system, can communicate to each other and form another (real or virtual) process as a result, like 'v', and then the vector 'v' can be considered for the newly created process. Vector 'v' is either in the same direction as the general state vector or not. If they are in the same direction, configuration of the HPCS must not change. Otherwise, execution of that process will change the configuration and it needs to be reconfigured.

Generally, operator '#' represents the direction of the process created as a result of the interoperation of two processes, and in fact describes a concept called the effect of sub-activities on the general state vector. However, it makes the decision about the direction of the obtained vector with the general state vector.

According to what is called customized configuration of the HPCS, given that both processes 'i' and 'j' existed explicitly at design time of the HPCS, the system had been configured by the HPCS expert such that both processes reach their required performance. However, the interoperation of these two processes may not have been accurately mentioned at design time. The vector 'v' can be thought as a virtual (e.g., the interoperation of two processes does not conclude in a process to be created in the system, but some effects are exerted on the HPCS) or real process.

If the effects of process 'v' on the HPCS were mentioned explicitly at design time, the HPCS experts must have taken into account the necessary configuration for executing process 'v' with its required performance. Otherwise, if process 'v' was not explicitly mentioned at design time, it shows the dynamic nature of the interoperation of processes 'i' and 'j'. The AMRC model, based on operator #, creates process 'v' within an HPCS machine and gives the following definition:

$$\begin{aligned}(\text{Machine } X) : (v_{IO} \text{ compare } I_{k_{IO}}) \text{ and } (v_F \text{ compare } I_{k_F})\\ \text{and } (v_M \text{ compare } I_{k_M}) \text{ and } (v_P \text{ compare } I_{k_P})\end{aligned} \tag{12}$$

Equation (12), the difference of vector 'v' and vector I_k, represents how much the newly created process 'v' is in the direction of the problem's solution. Let us assume that the vector of the created process forms an angle with the general state vector I_k. This means that the requirements of process 'v' must be handled such that this angle is removed.

Equation (12) is used by the HPCS management system in each machine to be able to decide for occurrence or non-occurrence of distortion. In Eq. (12), the operator

"and" means that HPCS management system should check all these comparisons and in case the condition of I/O condition vector, memory, file, or processor and inter-process communications related to machine x is changed compared to global operation condition vector, then a distortion condition is created inside machine x. In other words, Eq. (10) pursues discovery of the fact that whether machine x is or is not afflicted with one of the distortion models because of executing various processes. Pseudo-code 2 can be used for Eq. (12) inside each machine that is a member to HPCS:

Pseudo-code 2 General schema of Eq. (12)

```
In kernel level after execution of the system call:
If Process v such that involved HPCS operation do activity {
If system call is I/O then {
        Update vector v_io
        If (Compare v_io with I_io) is not True then
                Send I/O state and run AMRC Function;}
If system call is File then {
        Update vector v_f
        If (Compare v_f with I_f) is not True then
                Send file state and run AMRC Function;}
If system call is memory then {
                Update vector v_m
                If (Compare v_m with I_m) is not True then
                        Send memory state and run AMRC Function;}
If system call is Process then {
                Update vector v_p
                If (Compare v_p with I_p) is not True then
                        Send process state and run AMRC Function;}
```

As stated, a heterogeneous machine in the system has some advantages with respect to the others in executing some operations. This relative advantage is caused by one of the four factors defined in T2 principle. The AMRC reconfiguration model can solve the problem of initial configuration being altered using three operations: first, a simple comparison, almost a comparison between the information stored in two data structures; second, the heterogeneity concept in HPCSs; and third, the relative advantage of each machine with respect to each of the four elements defined in the T2 principle.

Now we should calculate the direction of vector 'v'. The first and most important question about operator # concerns the manner of calculating vector 'v' using it. When an HPCS uses the AMRC model to determine the element that is not in the same direction as the current configuration of the HPCS and investigates the factor of this matter, a data structure called *HPCS_Vector* is created in the system. This data structure subsequently contains the four sub-data structures: *HPCS_Vector_IO*, *HPCS_Vector_F*, *HPCS_Vector_M*, and *HPCS_Vector_P*. On the other hand, for each process created in the system at design and execution phases, a data structure

called *Process_Vector*, containing itself four sub-data structures, is created. These four sub-data structures are exactly the same as those created in *HPCS_Vector*.

Operator # creates vector 'v'. If the result of the interoperation between processes i and j causes a new problem name $Problem_v$, then $Problem_v$ is considered a sub-problem of the main problem, which the HPCS is executing.

Based on the AMRC model, when a sub-problem is created inside each machine, then as viewed by the HPCS management system, the sub-problem based on Eqs. (9) and (10) turns to a duodecimal sub-problem describing vectors. It should be noticed that the above sets are considered as vector sets in this situation, and for each set a state, called final state, can be defined. To better understand, let us assume that the problem to be solved by the HPCS is changed to twelve $Problem_v$s. Each $Problem_v$ is a finite set and is completed under a special condition. This means that by the occurrence of each activity 'k', this vector set is either moved toward or away from its completion.

Now, let us assume for process 'i', there exists a shadow of vector I_k, called 'S_1', and for process 'j', a shadow of vector I_k, called 'S_2'. These two vectors are considered in the customized configuration of the system, and on the other hand, they are members of Saurus. Therefore, operator '#' is defined for them. Let us assume that 'S' represents the result of applying operator '#' on the two vectors (it represents I_k regarding sub-activities of process 'v'). If sub-activities of 'v' are in the same direction with the direction of 'S', then they are also in the same direction with the direction of I_k and vice versa. In such a condition the following equation holds:

$$\forall k \in \ddot{S} \text{ so that } S = \ddot{S}\#\dddot{S} \text{ then } S \xrightarrow{\text{Toward finding}} \overbrace{S} \text{ or } S \xrightarrow{\text{Toward finding}} \breve{S} \tag{13}$$

Equation (13) describes how when an instruction is executed by a process, the HPCS management system in the local machine investigates whether or not the corresponding vector with the executed instruction is stabilizing toward the equivalent global operation with its own vector. When an instruction is executed by a process, in view of the HPCS management system, the most important question would be whether executing the instruction has caused a distortion in HPCS. To investigate the issue, the HPCS management system compares the vector resulted from changes caused by instruction execution to an equivalent vector of global operation (one of the problem describing vectors); if the direction-weight of the vector is resulted from changes aligned with the vector weight-direction of equivalent global operation, then executing the instruction shows lack of occurrence of distortion on configuration level of HPCS.

Set '$\ddot{S}$' can be any one of the twelve sets above, and it is the current state vector of the set. According to the property of operator '#', since both '$\ddot{S}$' and '$\dddot{S}$' are members of the Saurus, applying operator '#' to '$\ddot{S}$' and '$\dddot{S}$' causes state vector 'S' to move toward either state vector '$\overbrace{S}$' or '$\breve{r}$'. State vector '$\overbrace{S}$' represents all the necessary events occurred for set '$\ddot{S}$' to complete in the HPCS. On the other hand, state vector '$\breve{v}$' is a zero state vector and indicates that no activity toward completion of set '$\ddot{S}$' has been performed. Therefore, Eq. (13) states if activity 'k' is of type set '$\ddot{S}$', then the state vector 'S' moves either toward '$\overbrace{S}$' or '$\breve{o}$'. However, state vector 'S' in the case of any activity is vector I_K.

7.7 Operator Δ in the AMRC model

The AMRC model assumes the occurrence of two general events in the system causes a process with dynamic nature to be created in the system. Another situation under which dynamism resulted occurs when a process creates another real or virtual process with dynamic nature which had not been considered in the customized configuration of the HPCS. However, in this situation, creation of a process is a result of a change in the value of one of the elements of the set of resources (such as CPU power, capacity of the main memory, network latency, etc.) with which the creator process is interoperating.

Operator 'Δ' describes the state of processes with dynamic nature. We assume that "Heterodont" and "Saurus" are bidirectional structures in Cartesian space. However, in practice, the aforementioned structures can be defined in any multidimensional space.

As mentioned before, generally a program can be divided into four general parts. Each part has some special features and can be considered by these features: for example, the processing part has features such as CPU time, CPU usage, and so forth; the memory-based part has features such as allocated memory size, shared memory size, etc.; the disk-based part includes page fault ratio, seek time, amount of read and write, etc.; and finally, the I/O-based part has features like I/O delay time, access time, queue size, etc. All of these features form the Heterodont set. Based on these two set, Heterodont and Saurus, operator 'Δ' is defined. Equation (14) defines the operator 'Δ' for every two members of "Heterodont" and "Saurus".

$$\forall i \in \text{Heterodont and } j \in \text{Saurus: } v = i \,\Delta j$$
$$\text{so that } v \text{ presents the role of process } j \text{ in sloving a given problem} \qquad (14)$$

Equation (14) assumes that the state of a process is equal to vector 'i' before the value of the HPCS set of resources element executing the process is changed. Then, according to operator 'Δ', process interoperation and its state is set to process 'v' after the scalar value (amount of available resources) is changed to 'j'. Using an approach like the one described for operator '#', direction of vector 'v' is calculated. This calculation is carried out to decide whether process 'v' is in the direction of the configuration of the HPCS or not. When the dynamic nature is initiated by the operator 'Δ', the HPCS management system has the choice between two approaches for decision making. The first approach suggests changing the scalar value (resource state) to its previous state, and the second one recommends using the concept of relative advantage for changing the location where process 'v' is being executed to a suitable location.

8 Discussion

The presented model, AMRC, for HPCSs reconfiguration at runtime; first offers the mathematical definition for HPCS based on generating spaces, and second, presents a mathematical definition of the executive element (Process) based on vector algebra

derived from generating spaces of HPCS. The mathematical definition of HPCS and executive element has resulted in the fact that the HPCS management system, against considering the process as an element that is defined based on load distribution, maps the process functional space to a space called the problem describing duodecimal vectors space (global operation).

Defining each process as the executive element is based on the global operation and the generating spaces of HPCS. On the other hand, in the AMRC model, the HPCS management system (contrary to traditional systems) does not use fundamental unit of load distribution to achieve its goals, but it considers each process as an executive element that is on a beneficiary element of HPCS, and a set of activities should be performed to accomplish some part of the global operation. The functional definition of process element is based on Eq. (10). That fact shows that the local management system maps the process functional space to the space of the problem describing duodecimal vector space.

In the traditional systems, when the HPCS management system is identifying, describing, and analyzing a process, it considers the process definition based on load distribution fundamental activity, as well as considers the corresponding role of the process as the criterion to describe the process, and all the definable activities that are executed on the process are based on load distribution. In the AMRC model, because of mapping the functional space of the process to the space of the problem describing duodecimal vector, the HPCS management system does not consider the processes based on pivotal load distribution fundamental unit as the criterion for investigation, identification, and description of it.

If we consider two spaces of the HPCS management system and the process, then the mapping of the process space in the HPCS management system (to identify the process) is what resources are being used by the process and on the basis of what load distribution is the process using that resource, and when resources required by the process are challenged somehow, how does the load distribution model cope with the challenges? In fact, the local distribution manager maps the process to its own space and considers any challenge of load distribution general policy as the process challenge, and it considers any mechanism of load distribution policy as the mechanism for coping with the challenge.

However, in the AMRC model, the HPCS management system considers each process as an element through which execution of each instruction results in changes (weight or direction) in the problem describing duodecimal vectors; therefore, mapping the processes to the HPCS management system space is what changes are made in the sub-problem describing duodecimal vectors by executing each instruction by the process. This makes the HPCS management system (against using various mechanisms for coping with various distortions) use the unique mechanism of generating space of HPCS based only on two functions of #, Δ to determine the changes of the sub-problem describing duodecimal vectors. In order to remove the changes caused by executing the instructions that cause distortion via processes in the local machines, the HPCS management system changes the generating spaces of HPCS. The HPCS management system uses the model of decreasing or increasing the generating spaces effects on a vector to change the generating spaces of HPCS. Thus, it is necessary to note that when a new vector (process) is developed based on operators #, Δ, if the vector is not in alignment of weight and direction of its equivalent vector in the problem

describing duodecimal vectors (created in starting machine of global operation), then the HPCS management system corrects the vector direction and weight by changing the amount or effect of one or several H, SS, AS, or F generating spaces, so that the direction and weight of the vector resulted from operators #, Δ, always align with the corresponding vector in the problem describing duodecimal vectors.

9 Conclusion

For modeling the reconfiguration of HPCSs at runtime, the paper gives a definition for dynamism, which may be initiated by the dynamism of the problem itself or the dynamism of the underlying platform. The AMRC model introduces two operators for recognizing them by using vector algebra concept based on the situations causing such processes to be created. The AMRC model uses vector algebra to model the dynamic process whose direction is not the same as that of the customized configuration of the HPCS based on the vector concept and provides this possibility for HPCS management system to make a decision about whether the state vector of the process is in the direction of the configuration of the HPCS or not. Furthermore, by investigating the angle formed between the state vectors of the problem's processes and the general state vector, the needed changes in order to return to the initial customize state are specified. One of the main features of vector algebra is that it is independent from time and place. This property of vector algebra allows HPCS experts to achieve runtime-based HPCS with the ability to expand in the HPCS configuration. The implementation of the AMRC model provides a customized configuration pattern at design time, a successful pattern in running problems is transferred to runtime, and dynamic problems in exascale duration utilize an HPCS with maximum customization at runtime. If fact, an HPCS based on the AMRC model can reconfigure itself at runtime due to changes occurring in the system and run the problems in customized configuration. For future direction, by extending the AMRC model, it will be possible to design an algebraic model for HPCS in every aspect and have an HPCS algebraic model.

References

1. Dongarra J, Sterling T, Simon H, Strohmaier E (2005) High performance computing, clusters, constellations, MPPs, and future directions. IEEE Comput Sci Eng 7(2):51–59
2. Gropp W, Lusk E, Sterling T (2003) Beowulf cluster computing with Linux, 2nd edn. MIT Press, Cambridge
3. Reyes S, Niño A, Muñoz-Caro C (2005) Customizing clustering computing for a computational chemistry environment. The case of the DBO-83 nicotinic analgesic. J Mol Struct (Theochem) 41–48
4. Dongarra J, Beckman P et al (2010) International exascale software project roadmap UT-CS-10-652
5. Sterling T (2009) The biggest need: a new model of computation. Int J High Perform Comput Appl 23(4):335–336
6. Brittan M, Kowalik J (2005) Autonomous performance and risk management in large distributed systems and grids. In: Grandinetti L (ed) Grid computing: the new frontier of high performance computing, vol 141, pp 225–253
7. Maccabe A, Falter H, Kramer W (2009) Resource management. Int J High Perform Comput Appl 23:347

8. Vecchiola C, Pandey S, Buyya R (2009) High-performance cloud computing: a view of scientific applications. In: The 10th international symposium on pervasive systems, algorithms and networks, Kaohsiung, Taiwan, December 14–16
9. Sarkar AD, Roy S, Biswas S, Mukherjee N (2006) An integrated framework for performance analysis and tuning in grid environment. In: International conference on high performance computing HiPC, India, December 18–21
10. Jacob AM, Troxel IA, George AD (2004) Distributed configuration management for reconfigurable cluster computing. In: International conference on engineering of reconfigurable systems and algorithms, Las Vegas, USA, June 21–24
11. Sass R, Kritikos WV, Schmidt AG, Beeravolu S, Beeraka P (2007) Reconfigurable computing cluster (RCC) project: investigating the feasibility of FPGA-based petascale computing, field-programmable custom computing machines—FCCM, pp. 127–140
12. El-Araby E, Gonzalez I, El-Ghazawi T (2009) Exploiting partial run-time reconfiguration for high-performance reconfigurable computing. ACM Trans Reconfigurable Technol Syst 1(4):21
13. Kogge PM, Bergman K et al (2008) ExaScale computing study: technology challenges in achieving exascale systems, Univ. of Notre Dame, CSE Dept. tech, report TR-200813
14. Brightwell R, Pedretti K (2011) A perspective on operating and runtime systems for exascale computing, scalable system software. Sandia National Laboratories, Albuquerque, NM, USA, University of Houston, April 15
15. Beltran M, Guzman A (2009) The impact of workload variability on load balancing algorithms. Scalable Comput Pract Experience 10(2):131–146
16. Rajkumar B, Cortes T, Jin H (2001) Single system image. Int J High Perform Comput Appl 15(2):124–135
17. Hoffman FM, Hargrove WW (1999) Cluster computing: Linux taken to the extreme. Linux Mag 1(1):56–59
18. Malloy RP (1987) Invisible hand or sleight of Hand-Adam Smith, Richard Posner and the philosophy of law and economics. Univ Kans Law Rev 36:209
19. Shaojun Z (2012) Analysis and algorithm of load balancing strategy of the web server cluster system, communications and information processing. Springer, Berlin, pp 699–706
20. Carino RL, Banicescu I (2008) Dynamic load balancing with adaptive factoring methods in scientific applications. J Supercomput 44(1):41–63
21. Barak A, Shiloh A (2012) The MOSIX cluster operating system for high-performance computing on Linux clusters, multi-clusters and clouds
22. Choudhury AR, George T, Kedia M, Sabharwal Y, Saxena V (2013) Method for improving the performance of high performance computing applications on cloud using integrated load balancing. US patent 20,130,031,550, issued January 31
23. Lieber M, Grützun V, Wolke R, Müller MS, Nagel WE (2012) Highly scalable dynamic load balancing in the atmospheric modeling system COSMO-SPECS+ FD4. In: Applied parallel and scientific computing. Springer, Berlin, pp 131–141
24. Devine KD, Rajamanickam S, Boman EG (2013) Combinatorial scientific computing for exascale systems and applications. No. SAND2013-3968C, Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States)
25. Hubbard JH, Hubbard BB (2001) Vector calculus, linear algebra, and differential forms: a unified approach, 2nd edn. Prentice Hall, New York
26. Asanovic K, Bodik R, Demmel J, Keaveny T, Keutzer K, Kubiatowicz J, Morgan N et al (2009) A view of the parallel computing landscape. Commun ACM 52(10):56–67
27. Watanabe M (2013) High-performance computing based on high-speed dynamic reconfiguration, high-performance computing using FPGAs. Springer, New York, pp 605–627
28. Sharifi M, Mirtaheri SL, Mousavi Khaneghah E (2010) A dynamic framework for integrated management of all types of resources in P2P systems. J Supercomput 52(2):149–170
29. Sharifi M, Mousavi Khaneghah E, Kashyian M, Mirtaheri SL (2012) A platform independent distributed IPC mechanism in support of programming heterogeneous distributed systems. J Supercomput 59(1):548–567
30. Churchman CW (1968) The systems approach. Delacorte Press, New York
31. El-Ghazawi T, El-Araby E, Huang M, Gaj K, Kindratenko V, Buell D (2008) The promise of high-performance reconfigurable computing. Computer 41(2):69–76

32. Van Craeynest K, Jaleel A, Eeckhout L, Narvaez P, Emer J (2012) Scheduling heterogeneous multi-cores through performance impact estimation (PIE). In: The 39th international symposium on computer architecture. IEEE Press, New York, pp 213–224
33. Marks M, Jantura J, Niewiadomska-Szynkiewicz E, Strzelczyk P, Góźdź K (2012) Heterogeneous GPU&CPU cluster for high performance computing in cryptography. Comput Sci 13(2):63–79