J Supercomput (2018) 74:1715–1764
https://doi.org/10.1007/s11227-017-2191-7

# A mathematical model to calculate real cost/performance in software distributed shared memory on computing environments

**Ehsan Mousavi Khaneghah**[1] · **Nosratollah Shadnoush**[2] · **Amir Hossein Ghobakhlou**[1]

Published online: 16 November 2017

**Abstract** One of the important factors in high-performance computing (HPC) is the cost/performance ratio. Calculation of cost/performance ratio is the main criterion for the separation of hardware computing systems (supercomputers) from software computing systems (Cluster, Grid, Peer-to-Peer). There are various economic methods to calculate hardware cost. In addition, there are numerous methods in software engineering to calculate the cost of developing and programming the scientific and engineering software. The computing power in the aforementioned systems is basically calculated with programs like LINPACK and HPCL. The inter-process communication is considered as a variable in calculating the cost of executing the scientific programs, whose nature and amount depends on the program execution itself. As there is a high dependency of effective variables in cost calculation of inter-process communications during the program execution, it should be used for calculating the cost of any application. This paper complements the existing methods by presenting a more comprehensive and accurate method to calculate the real cost of distributed shared memory (DSM) mechanisms used by HPC Systems. Therefore, a systematic method has been used to achieve a whole equation for DSM costing, determine the effective factors of the cost, and propose a method based on costing economic methods. Effective parameters are classified into two groups, namely DSM-inhere dependent and application-specific

✉ Ehsan Mousavi Khaneghah
EMousavi@Shahed.ac.ir

Nosratollah Shadnoush
Nosratollah.Shadnosh@Gmail.com

Amir Hossein Ghobakhlou
Amirh_gh@Outlook.com

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

2 Department of Management, Faculty of Management, Central Branch, Islamic Azad, Tehran, Iran

dependent parameters. Each parameter is presented and discussed, and the correlation between them specifies the system's weight on DSM real cost according to which the cost is modeled and validated analytically.

**Keywords** Distributed shared memory (DSM) · Cost model · High-performance computing (HPC) · False sharing · Correlation

## 1 Introduction

One of the important factors in high-performance computing (HPC) is the cost/performance ratio [1,2], which acts as an economic justification for running scientific program on HPC systems. At times, these systems can handle an execution of a scientific program, but they lack economic efficiency. This cost/performance ratio parameter specifies the type of scientific program that can run in HPC systems like Cluster, Grid, and Peer-to-Peer (P2P) [3].

There are varied ways to calculate this ratio. For instance, Zhang et al. [4] and Kurmann et al. [5] investigated this ratio in network interconnections to specify its importance in low-end networks in cluster systems. Rauber et al. [6] and Tootaghaj et al [7] also studied on how to obtain cost/performance trade-offs by developing a simple parallel benchmark model.

In HPC systems, some mechanisms are used for measuring the system efficiency [8–10], which challenges the patterns of measuring the system cost in computing the cost/performance ratio. The cost of a system is likely to be dependent on the efficiency and nature of the system that defines the methods used for the calculation of execution cost in computing systems like Cloud or Grid [11,12].

Calculating the cost of these systems is the main challenge in the cost/performance ratio evaluation. To understand the cost calculation of a system, the cost involved in the execution of applications in HPC system should be understood followed by the mechanisms used to calculate the cost. After the completion of the survey, two methods or solutions have been formulated to calculate the cost in HPC systems.

The first solution is the mathematical model of calculating cost and efficiency of each scientific program or computing system management application or any special feature of the computing system [7,12,14]. This method is called as minor fee calculation because of its cost that is implemented on the basis of each specified element or content.

The most important feature of the minor fee calculation is the process of using program execution for the calculation of cost and the cost/performance ratio. In this method, the general mechanism of cost extraction is based on a scientific program (or specific program) that is run in a specified computing system. Some of the system elements or features cause the cost, which is calculated during the application of the running process. The most important advantage of this method is the feature of proposing the exact cost of a program execution in certain computing systems [1]. The focus of this method is to recognize the element or feature of the system that causes the cost creation during the program execution. In this method, each element or feature causing the cost creation is surveyed independently, which gives a precise vision on why the certain feature causes cost creation. This method not only helps in understanding the communication of program, system, and cost but also the behaviors of the feature of cost creation in different runs. Calculation of each factor of cost creation independently causes a lack of vision on their influence on each other. The most important challenge is that there is no general pattern in the survey process of these factors or definitions of cost creation. If the cost creation parameters, obtained from the scientific programs, changes then there will be no certain decision about the reuse of the obtained parameters.

The second solution is using a public pattern to calculate the cost and mentioned coefficient in computing systems [15–18]. In this public method of cost calculation in a computing system, the calculation is based on "a basic content," which can be a defined content in the system or by the user. Elements like the time of system used, system creation cost [12], power [19], and data can be considered as the basic content of cost calculation in the system. When an element is known as a basic content of cost calculation, it must be redefined based on an axial element in each system. The activities caused in each element or features in a system lead to cost creation, which is further redefined based on the fundamental activities defined for the axial element.

Based on the classic patterns of costing, except operational or fixed cost calculating method and variable costs, efforts are made to know the effective elements of cost creation in a computing system. As a result, the communication between effective elements on cost is specified. In cost calculation method, the most important content is clearly understanding and correct selection of the axial element.

Based on the fact that a classic cost computing pattern is generally used, the main activity defines the axial element. If not chosen correctly, the parameters of the axial element affecting the cost could not be considered. Moreover, the parameters that do not affect the costing of this element or define other elements might be considered as an activity during the survey. Hence, there is a need to understand the exact meaning of axial element and defining the other elements and existing activity in a system. However, defining the cost based on an axial element causes the exact pattern to calculate the costing that would redefine its activities and elements.

On the other hand, there is a need to have a clear knowledge about the cost of running a program in an HPC system and what elements are involved in it in order to calculate the exact cost. In a general overview, the cost of executing a program in HPC system is followed by the system creation cost, designing cost, scientific software development cost, system management software cost, and the cost of the mechanism that is used for converting the scientific program to a parallel and distributed program [1]. There are many economical ways for calculating the cost of a system creation, even the cost created by the depreciation of system is included. On the other hand, the development cost of scientific software can be calculated by the existing methods in software engineering. The cost of purchasing or developing the system management software can also be calculated in this way. Calculating the cost of the mechanism of converting a scientific program to a parallel and distributed one, a different approach is required. This involved the nature of the cost depending on the execution of the process of global activities during the running time, and it shows the cost of process distributed in the computing systems.

For calculating the cost of inter-process mechanism in computing systems, it should be enabled to answer the questions like ‘What kinds of dependency are present in between the processes? How does the displacement and distribution of processes occur in running time? Moreover, the critical question is ‘What kinds of faults affect the cost calculation during the distributed and parallel execution of processes.’

One of the most important questions that should be answered while calculating the inter-process cost in computing systems is the cost calculation pattern. In this paper, the public pattern for calculating the inter-process communication (IPC) cost is used for which there is need to consider the method as a cost calculation benchmark among the other methods of IPC.

Distributed shared memory (DSM) and message-oriented middleware can be used as two mechanisms for the distributed IPCs to convert the program from a scientific one to a distributed one. In this paper, the main focus is on DSM mechanism, which provides a probability of IPCs between remote processes as well as the local shared memory.

If the cost of running the scientific program on HPC systems like Cluster, Grid, and P2P Systems are to be calculated, there will be a need to calculate the cost of DSM mechanisms used to setup the HPC Systems.

In one of the solutions proposed by Kaplan et al. [20], the cost of a system is calculated on the basis of its activities and by observing each activity. The cost of its effective elements is derived and calculated to find the best combination of its effective elements and improve the costing [1].

The other popular solution is analyzing the elements of the system based on their functionality with the approach of calculating their execution cost imposed by the system along with the cost effectiveness for each element [21,22]. This solution is popular in management, decision-making, as well as its decomposition and analysis. The most popular solution is based on the system synthesis and has a special approach to complex issues as sets of variables and their reactions. The main difference of this solution with the previous one is related to its attention to reaction and interaction of the elements. For instance, Berriman et al. [23] tried to survey the cost and efficiency contents in Astronomy software in Cloud computing systems. In this paper, each activity that causes cost creation in the execution of the scientific program has been calculated for each program. The cost calculation in this paper is performed to know the effective factors on cost per specific scientific program in a specific computing system. In the study by Deelman et al. [24], information transferring cost has been considered as the only parameter in a computing system. In Expósito et al.’s [1] paper, the operations execution cost has been considered in Cloud computing systems and also in the scientific programs. In the study by Han et al. [13], the cost of using the computing system per special kind of scientific program has been surveyed, and the costing mechanism has been used in finding a solution for allocating the efficient resource. Zhai et al. [25] introduced the cost of execution on Cluster systems and Cloud computing systems as the model used for costing. Therefore, the general mechanism involves the cost calculation of the program execution on Cluster and Cloud system. There is no public model for calculating the cost, but the cost of each element or feature causing the cost creation during program execution has been calculated independently.

In the studies by De Alfonso et al. [19] and Woitaszek et al. [26], an economical method is used for calculating the cost of the computing system. In De Alfonso et al.'s [19] study, a public method is proposed for calculating the energy cost for users to transfer from Cluster computing systems environment to Cloud computing systems. The most important advantage of this method is the consideration of a public model to assess the effective factors in the cost creation of Cluster computing systems. For the calculation of this cost, the general methods have been used in consideration with the fixed and variable costs. The effective elements on fixed and variable costs have also been proposed about energy consumption, which has also been used in Cloud computing systems. The used costing method is a breakdown of fixed and variable cost methods. In this method, it is supposed that the total cost is equal to the fixed and variable costs, and the solutions use the economic pattern to evaluate the cost benefit of the public pattern of cost calculation. In this paper, the public pattern and classic models have been used to calculate the cost of DSM.

Based on the synthesis solution for cost calculation of the cluster, the hardware, system software, and application are the three cost generator sub-systems. Hardware and system software cost is obvious in the system because the provider of the cluster system pays it directly.

In this paper, a mathematical model is presented to calculate the cost of DSM instructions as a global model based on the synthesis solution. Considering the DSM program as a systematic program, old methods were used for the calculation of the cost creation and DSM mechanism.

DSM has two essential characteristics, i.e., hiding the complexity of the underlying platform (transparency) and limitless usage of the beneficiary machines in executing the instructions on it. In the traditional programming pattern, programmers focus on the scientific program theories, their details, and their complexities by using local memory. The memory manager controls the read, write, consistency, and data movement between the processes and operations. In the DSM pattern, programmers only focus on the challenges of parallelism concept of scientific programs to develop efficient software programs [27]. By this description, the cost calculation of DSM instruction is a significant challenge in scientific computing systems like Cluster, Grid, and P2P systems. Therefore, a mathematical model is presented for calculating the real cost of DSM based on the synthesis solution in computing environments. The functions involved in calculating the DSM cost for any scientific program and computing system can also be used.

In this paper, each parameter's relation with DSM real cost is obtained, and the amount of each parameter's influence on the cost is specified by a variable denoting the weight of the parameter, thereby representing the DSM real cost in an analytical manner. Satisfied function has been used for the calculation of the effective coefficient of each cost part on total cost. For extraction of the coefficient of each cost part, considering total cost as an independent variable, component cost as the dependent variable, and adopting statistic distribution as the mentioned function, efforts are made to define the satisfied function between dependent and independent variable.

Given the process of calculating the DSM real cost discussed in this paper, it is possible to choose the best strategy in different situations based on its cost. If the cost of other IPC mechanisms is available, it is possible to compare it with the DSM real cost to choose the mechanism with the lower cost. There is a metric approach in this paper, and the principles and methodologies of accounting and management sciences are used to calculate the real cost [28,29].

In biological systems (P2P and Grid), determination of the communication mechanism used between the present system and added machine by Resource Discovery is a critical subject. The use of IPC mechanism results in the cost creation of the system. When Resource Discovery tries to find a machine in a network (which has the required resources), the system tries to create IPC between the system and the machine. If there are no calculating tools for the cost of using the IPC mechanism in this system, the cost/performance coefficient will not be meaningful.

In this paper, cost accounting theory [30,31] is used as a model for the calculation of the real cost/performance in software DSM on computing environments. Cost definition is based on cost definition [32] and resource consumption.

The second section presents the related works of DSM, followed by the third section that discusses the real cost of application specific dependent parameters. The fourth section presents the case study, and the fifth section concludes the findings of the paper.

## 2 Related works

Parallel processing has been widely used and enriched by HPC community over the past years in basically two categories, namely shared memory and message passing, at the developmental and programming levels. Shared memory can be accomplished using small-scale symmetric multiprocessors effectively, generating low-end parallel systems that serve as higher-performance systems [4].

Shared memory, however, is increasingly popular in the mid-range world. Sun, IBM, HP, and SGI are vendors that scale to a few hundreds of processors. These machines have significantly higher per-processor costs than smaller symmetric multiparallel (SMPs) ones. Consequently, to decrease computing cost, many companies prefer to deploy clusters of SMPs, with low-latency networks and high-bandwidth connections, when such replacement is accessible for their applications [33]. Communication and data sharing between SMPs in HPC (Like Cluster, Grid, and P2P) need IPC mechanisms, including message-passing mechanisms [34]. Nevertheless, programmers have to take care of data distribution across the system and manage complex communications due to different address spaces, process migration, and other issues [35]. The intermediate solution that combines the advantages of both of them is the best approach. Therefore, DSM mechanism is proposed in which message passing is hidden by avoiding explicit sending of messages between processes at the program level.

DSM mechanisms consist of different protocols, which have been developed with various implementation viewpoints. Due to the importance of the cost problem, some works found in the literature [33,36–42] specify whether distributed IPC or DSMs is more cost-effective in message-based systems. Some of these works estimate the cost of DSM based on the number of additional messages or the number of memory accesses.

DSM performance is dependent on the memory access behavior of the application. For instance, previous studies [40,43] have presented a cost model for a special kind of DSM that uses competitive-update protocols. These protocols are a kind of memory access that calculates the number of updates in each segment by a probability density function; these also calculate message-passing overhead and the cost of message passing of a base message size. They represent an adaptive DSM base and the competitive protocols required to minimize a predefined cost function, which is the number of messages and the amount of data transfer. Finally, they show that performance is improved by dynamically selecting a suitable protocol.

Studies [43–46] have shown that DSMs involve the use of average memory access costs, the number of nodes, sending packet costs, the probability of fault and ratio of reading and overwrite.

SMP clusters are cost-effective solutions to building large-scale parallel machines, but for fine-grain application, SMP clusters have poor performance and low cost/performance ratio [46–48].

Existing studies on estimating the cost of IPC mechanisms, including DSM, are either biased to specifics of their problem specifications or only account for less effective parameters on DSM real cost to make their calculations simpler, helping the less accurate calculations. All effective parameters are identified and described in details, and subsequently, the weights of their influence on DSM real cost as well as their influence on each other, i.e., their relations, are determined by using the normal distribution. Granularity is considered as an important effective parameter of DSM real cost. A coefficient named clustering coefficient is also presented.

Studies [49–52] have presented a middleware for the proposed DSM system in wide area networks. Musical et al. [53] proposed that the DSM uses three effective parameters for cost, namely component cost, agreement protocol cost, and update cost. The number of nodes, the number of requests, and system configuration influence cost update and are related to the sequential consistency model of the proposed DSM.

Kim and Vaidya [54] and Gray [55] introduced an approach to have highly available access to DSM data with low operation cost. Their research is based on a class of coherent protocols for DSM, named competitive-update protocol. They have divided requirement operations in this protocol to three parts, namely creation, read, and write operations; all these operations affect the cost. They tried to decrease the number of these operations by monitoring the workload of the system and adjusting certain protocol parameters to have minimum page data availability of DSM and induce a decrease in operational cost.

In a previous study [56], a model for analyzing the DSM cost has been proposed based on the axial element of competitive update. In this model, DSM pattern for data management is based on Segment. In this paper, probability density function is used for calculating the DSM cost.

The proposed model is a probable model for calculating the cost in DSM per occurrence of a special kind of update related to a segment between two creator machines of DSM. The most important advantage of this method is analyzing DSM cost based on an event which happens again and again in a system based on DSM, especially in a computing environment. The focus on special kind of updating means that all possible spatial modes in the system are being surveyed because of the occurrence of the events in the proposed model. The most important limitation of this paper is not surveying the other factors affecting DSM cost.

In a DSM system in which data is an axial element, there are other elements that create cost based on data and defined activities for the data element. The cost assessment mechanism of this paper is a conventional mechanism for assessment. The main assumption of the evaluation method of the proposed model is that the proposed model should describe the costs according to the real world in which the DSM system manager works.

A previous study [57] proposed a general framework for surveying the costs in a high-performance Cloud computing system. The cost extraction pattern in this method was based on the public pattern. The proposed framework included five elements related to the cost of using Cloud computing systems; based on these parameters, a prediction is proposed. The used pattern in this method can be used in any computing system considering the special features of the computing system. The most important advantage of this method is that the proposed analysis is based on recent information and creation of knowledge-based structure. In framework architecture, based on the present mode and obtained information, a prediction is proposed about the cost of using the resources. The most important challenge of this field is that no entry is related to the cost of creation, management, and maintenance of HPC systems. The reason is Cloud computing system. Anyway, the main mechanism of this method is that the cost of using HPC is based on pay-as-you-go, and there is no surveying of the elements related to costs in the system; the only thing proposed is the prediction framework about the cost of used resources.

There are some solutions to calculate the cost of applications such as using a line of code (LOC) metrics [58] and assigning a cost coefficient to each line based on its complexity. Indeed, this solution is based on decomposing and analysis.

There are two fundamental problems in using LOC metrics for DSM. First, the assigned cost coefficient to each line is considered individually based on its execution complexity on the processor without any attention to the relation between the lines of codes and their reaction and influence on each other. The second problem is related to the lack of attention to global operations in executing the applications on the cluster system. There are some parts in Cluster-based applications in which the global operations are executed. Calculating these lines' complexity is difficult due to the dependence of lines execution on the machine. On the other hand, deriving the complex nature of these codes has a direct relation to their transparency, and by increasing their transparency, the complexity of calculating the hidden cost increases. Therefore, due to its complexity, the decomposing and analysis approach is rarely used for calculating the cost. In contrast, synthesis is used more often as a method for calculating the cost.

One of the used mechanisms in computing systems for calculating the IPC mechanism cost is USD per GOPS [1,59]. It means the cost of Giga Operation per Second execution. This cost can be calculated in systems in which the process units are rented by the user. In this method, the cost of process unit execution is considered in specific period than the number of operations in time unit as cost criterion of using the process units. This method is the minor cost calculation method. The specific factor of this method is the cost of using the CPU. CPU using cost and the number of executable operations are two units which are specified by CPU. These two elements can only calculate the total cost of CPU and cannot calculate the details of cost and details of computing program execution cost. Therefore, they are used for the computing systems that are not created and maintained by the customer. The customer in such a system can only rent the computing system in order to run the specific program (or programs). So, if the creator of the system wants to extract execution cost of computing program, USD per GOPS content will not be used.

## 2.1 Review of distributed shared memory based on system approach

Since the cost of DSM is due to different machines and elements of the system and because of DSM transparency, it is not possible to clarify the role of each element in increasing the cost of DSM. Therefore, a systematic approach is needed to know the real cost of DSM to give an integrated viewpoint.

There are two approaches to understanding the problems: reductionism and system approaches [60].

In the system approach, the focus is on arrangement and relation between the parts and how they work together as a whole. This approach in modeling the cost of DSM gives some advantages too; *the first one* is the independence of the model to specific implementation and details.

This model can be applied to specific implementation by changing the coefficients as it is based on DSM mechanism. *The second one* is deriving the fundamental and general operations of DSM and applying the influence of them in the cost model of DSM. *The third one* is the consideration of the correlation of internal system reactions. *The fourth* is the consideration of the effects of hardware, system software, and applications on DSM cost.

The important problem in this state is finding a classic system to adapt DSM as a basic system and describing DSM costs. According to the nature of DSM, there are different systems such as supply/demand that are considered as the basic system. However, by investigating the nature of DSM, it can be concluded that DSM acts as a production line [61,62]. There are producer processes that give some data to DSM to transfer, and there are consumer processes that use the data. Figure 1 shows the logical and simple view of DSM as a production line. Therefore, DSM can be assumed as a production line system to get a cost model for it.

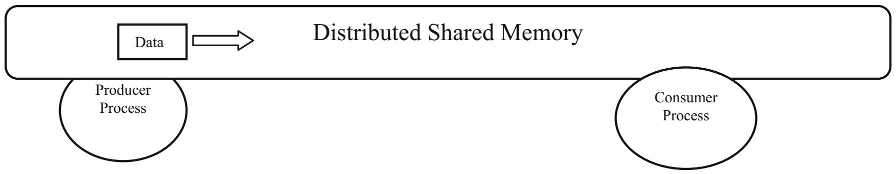

**Fig. 1** Distributed shared memory definition based on production line

As shown in Fig. 1, a one-to-one relationship between DSM in HPC and production systems can be without loss of generality.

In this paper, based on the system approach, the cost-effective operations in DSM are categorized so as to cover all fundamental operations of DSM and reach an accurate cost model.

Regarding this relationship, a decision about two issues can be made. (a) Which general parameters are considered as effective parameters in the cost of production system? Moreover, how these parameters affect each other and environmental parameters. (b) Some parameters, considered as basic parameters of the system, i.e., caused by the basic purpose of the system, are produced, and their impact on the real cost of using DSM is obtained.

### 2.2 General parameters of cost in DSM system based on system approach

Operational costs and activity costs [30] of systems are two parameters of the production line. Operational costs are set by costs involved in enabling a system to perform duties, which were determined in its purpose. As it is predictable, the costs of running activities in a system are called activity costs.

There are more exact patterns to differentiate between operational costs and activity costs. There is a general definition for operational costs that should be redefined by the system.

Operational costs of a system include sets of its creation, its maintenance, and execution of operations in the system. It is necessary to notice that the third set of operational costs refers to what cost should be paid for running activities in the system. Activity costs refer to what costs might be raised by the system while running activities.

Each of these costs is defined based on the parameters and variables of the system. One of the most important features of operational costs is to recognize relations between parameters creating cost in a system with each other and with system's environment. On the other hand, inherently activity-based costs are to a great extent dependent on activity, the basic purpose of the system and the set of activities that should be run in the system to reach its final goal.

Some effective parameters on DSM real cost are considered and divided into two general groups, namely DSM-inhere dependent parameters whose details have been published in a previous study [63] and application-specific dependent parameters. However, the application-specific dependent parameters are investigated with more details.

On the other hand, these sorts of costs are operational costs of DSM. Based on the communication between operational costs and activities costs, the nature of operational costs will be corrected after determining the operational costs. On taking a look at the nature of operational cost, it is obvious that operational costs must (A) communicate with the activities of the system in running time and (B) these costs must affect operational costs.

As mentioned before, there are two general groups of costs defined in DSM systems based on the production line model. These groups are named operational costs of

DSM system or costs caused by the distribution of the shared memory system (same as production system) and activity costs caused by execution of application programs on DSM system.

It should be noticed that the final goal of a DSM system is the execution of scientific programs, which need memory-based out-of-machine IPCs. Execution of such programs on DSM causes some costs, which is in line for reaching the final purpose of the system.

In a previous study [63], it was shown that DSM production and usage costs in HPC systems are categorized into three general levels in which their variables and reactions to each other and system-environment issues have been introduced. DSM establishes memory pattern for out-of-machine IPCs in HPC systems. Therefore, while an activity is running on DSM, some costs are formed in DSM system, due to inherent behaviors of program and usage of memory pattern.

The structure of operational costs for DSM (when this system is equivalent to production line system) is very specific and explicit. It relates to costs of creating the DSM (by the programmer or HPC manager) and maintaining it for transferring data between producer process and consumer process.

As mentioned earlier, activity costs do not use a specific pattern like in the case of operational costs. So, the equivalence of operational costs and production line triple costs for each system that is equivalent to production line system is expected. When a system like DSM is equivalent to the production line, there is no requirement to consider one-by-one activity costs of each system because the activity cost is based on basic system activity.

The most important thing about activity costs is understanding the basic activity for which the system has been created. In DSM, basic activity is based on shared memory, so there is an expectation that this system's activity costs is created based on the distributed memory activities of this system. On the other hand, these costs must be under the effects of the basic operation of this system (transferring data on DSM) [64].

If the basic activity of DSM system is changed, the effective elements on activity costs will change. If basic operation and activity of DSM system is changed from transferring data to any other content, the activity costs of the system will change.

## 2.3 Real cost of DSM-inhere dependent parameters

The logical model used by a programmer for implementing DSM is common because there is available memory equal to the total existing distributed memory in the system. It is proved that reaching this goal is impossible when using multi-computers systems because of the existence of various and complex challenges and limitations such as the establishment of persistent transparency. Hence, for the implemented DSMs to have a correct implementation of the DSM concept, it is required to proceed with DSM during the request until the expiration of the operations while transferring requested data from one memory to another, maintaining consistency. All such operations are costly.

This has been considered under the title of the parameters originated from the nature of the DSM [43].

In multi-computer systems that use DSM mechanism, the information transfer cost is much dependent on the size of data to be exchanged. Therefore, practically, a simple data transfer costs the same as a complex data structure transfer. Current DSMs are implemented using different protocols and software in various levels of operating system architecture. Therefore, the cost of transferring information in HPC systems that use DSM is neither equal to zero as in local programming models on standalone systems nor is a function of the amount of transferred data as in distributed memory programming models on distributed systems.

The mentioned costs in this section come to play when DSM is implemented as a software application at low levels. The implementation level is a critical condition because all defined costs in DSM production line make sense if the execution of an operation is hidden from users and handled by DSM management system and operating systems involved in the implementation of DSM.

The parameters affecting DSM real cost model are classified into two groups. One of these groups originates from the nature of DSM (as mentioned, these variables caused by the adjustment between DSM system and production line and operational costs).

### *2.3.1 DSM-inhere dependent parameters*

- Average Data Transfer Cost
- DSM Creation Cost
- DSM Maintenance Cost

In continuation, after checking the proposed costs in [63], there is an analysis of those costs in distributed computing systems like Grid and P2P computing systems for DSM-inhere dependent parameters.

I. *DSM creation cost* The DSM cost is the result of the parameters of some distributed memories at lower layers and the cost of protocols setting up the DSM. Due to the costly nature of such operations, the cost of DSM set up was assumed to be equal to the possible lowest level [63].

In the calculation of DSM cost, DSM is only assumed to run once at the beginning of execution of a program, remains during the program run and is destroyed upon completion of the program. Thus, the cost of establishing DSM in any situation is calculated only once forever.

What is the HPC system management protocol? This is the protocol used in distributed HPC systems to implement DSM without considering the information of transmission protocol. The protocol cost is equal to the cost of execution of the HPC system management software as per a certain task and set up by the creators of the HPC system management software.

The relation of these parameters is made dependent on the number of machines (as cost relates to numbers) composing the HPC system that uses DSM mechanism. So, the DSM creation cost is calculated as follows:

$$\begin{aligned} &\mathit{DSM\ Creation\ Cost} = \mathit{Average\ Run\ Cost} \\ &* \left[ \begin{array}{c} \mathit{Average\ Control\ Message\ Cost} \\ (\mathit{Number\ of\ DM} * \ \mathit{Average\ cost\ of\ each\ unit}) \\ + \sum_{i=1}^{N=\mathit{Max\ Machine\ in\ Global\ Activity}} \ \mathit{Cost\ of\ HPC} \\ \mathit{Management\ Software\ based\ on\ DSM\ license\ per\ node} \\ +\mathit{Cost\ of\ HPC\ Setup\ per\ node} \end{array} \right] \end{aligned} \tag{1}$$

Equation 1 has developed the format of the created cost of DSM which has been proposed in paper [15].

In Eq. 1, the run average cost per bit is the average cost of the program running on distributed memory. The number of DM is the number of distributed memories from the DSM. The administrator based on the cost of software calculates the cost of HPC management software per execution. It should be noticed that the mentioned cost shows an impact of the environmental parameter, i.e., HPC system management software, and some effective hardware elements in HPC system on DSM cost.

In Eq. 1, DSM creation cost is the variable of cost, and the average cost of each unit and cost of HPC management, software based on DSM license per node, average run cost, average control message, cost of HPC setup per node are all cost variables. Among all the cost variables, the average cost of each unit has fixed cost in distributed systems, and other cost variables have dynamic cost. The scalable capacity of distributed computing systems is the reason of this content. In HPC systems, dynamic costs are changed to fixed costs.

For the development of using the content of created cost of DSM in computing systems that use pattern of biological systems like Grid, Cloud Computing, and P2P systems, global activity [52] content can be used.

When global activity forms in the system, the creator machine of global activity uses a centralized pattern for global activity if the mechanism of IPC of global activity is equivalent to DSM. Therefore, a DSM is created between running machines of global activity. So, the cost of DSM creation is calculated once. However, if the global activity creator machine uses decentralized or distributed pattern, a DSM is created per each communication in each local machine in which global activity happens. Therefore, the cost of DSM creation will be per each communication event.

In Eq. 1, there are two activity fixed costs: HPC management software based on DSM license node and cost of HPC setup per node. This cost is based on which parts of DSM manager are installed in which machine. One of the efficient methods for calculating the first cost is using the content of module or function; generally, DSM management systems follow two contents. In this situation, the cost of Module or Function related to DSM is calculated, and based on the kinds of module or functions that are installed on each machine, the cost of each machine can be calculated.

About the second cost, it is important to have some configuration in the machine when a machine wants to be a part of a global activity in the cluster, Grid, or P2P system. The execution of this configuration in each machine is considered under the supervision of HPC Setup per Node. One of the solutions is using the concept of time value. In this method, the cost of the required time for configuration of the machine for using in a computing system is considered [8,48]. In a cluster system, the cost of

HPC setup per node is fixed per total time of cluster activity and can be calculated in the cluster execution. But for computing systems like Grid and P2P, this cost is per each global activity in which the member machine of the computing system plays a role.

II. *DSM maintenance cost* A series of protocols are used for HPC and shared memory management to maintain the DSM [63].

Without even considering the information transmission protocols at multi-computer systems, the volume of communications and the exchanged messages to control DSM and the transferred data volume in the DSM are a series of factors that directly affect the maintenance cost of DSM. The DSM maintenance cost is calculated as follows:

$$\begin{aligned} &\mathit{DSM\ Maintenance\ Cost}\\ &= \left[\begin{array}{c} \left(\sum_{i=start\ DSM}^{Kill\ DSM} \mathit{Control\ Message}_i\right) \\ * \left[\sum_{i=Start\ DSM}^{Kill\ DSM} \left(\frac{\mathit{data\ transfer}_i}{\mathit{Page\ Size}_i}\right) * y1_i\right] \\ \left[(\mathit{Cost\ of\ HPC\ Management\ Software\ based\ on\ DSM\ license\ per\ node} \right. \\ \left. *\mathit{Sinking\ cost\ of\ each\ node}) * y2\right] \end{array}\right] \end{aligned} \tag{2}$$

Equation 2 is the extended format of the created cost of DSM, which has been proposed in a previous paper [63].

Here, $y_1$ is the weight coefficient to determine the importance of data transfer size cost, and $y_2$ is the weight coefficient for which the administrator should calculate the cost importance of control message size.

In Eq. 2, the maintenance cost is specified based on the execution cost of a single time of the HPC system management software, especially associated with DSM management.

In Eq. 2, maintenance cost, cost of HPC management software based on DSM license per node and sinking cost of each node are the cost variables. On the other hand, in Eq. 2, data transfer and control message are considered as fixed costs. If the number of code(s) can be fixed, all cost variables are fixed and otherwise are dynamic.

In Eq. 2, it is assumed that the computing system management element tries to create DSM for executing a global activity (the computational activity consists of more than one computing element); during the global activity, running DSM management tries to pay the cost of maintenance of DSM. The nature of this cost is different from the cost of data transferring on DSM. This cost is for maintenance of DSM for a specific global activity, which is paid by the DSM manager.

This cost is completely dependent on data transfer and control messages. In Eq. 2, the beginning of the creation of DSM to the end of the related activity is being surveyed; it is probable that the mechanism of DSM may be used in biological computing systems like Grid and P2P systems. In addition, the following scenarios should also be considered: (a) probably one machine is used in more than one global activity, (b) in execution of the global activity, there is a possibility that DSM needs to be created more than once and for different parts of the global activity.

With respect to Eq. 2, if data are transferring, the importance coefficient of DSM is getting higher and the y1 coefficient is also getting higher. Increase in transferring importance coefficient means the creation of special-purpose DSM or increase in control messages. The amount of importance of the coefficient can be equivalent to one hundred when DSM is transferring a hard real-time message and can be equivalent to one when there is a standard message. Y2 is also the coefficient importance of control messages in DSM management software. Depending on the importance of control mechanisms of DSM, the mentioned importance coefficient will increase.

III. *Average data transfer cost* In this paper, various DSM systems and their costs have been analyzed to find data transfer cost [63].

The data transfer cost is known to be directly related to the level of implementation of the protocols generating the DSM; this is a different amount on different systems. So, the implementation of the protocols is an important part of a DSM, which can lower the data transfer cost. However, it is possible to consider [0, 4096] as the basic data transfer for the systems.

Page size in DSM is usually deemed the same as what is typically considered in operating systems. In case the underlying operating system has a different memory pattern than Linux, the high limit is equal to the output of *getting confPAGESIZE* or the equivalent command. However, in this case, the high limit is chosen to be 4096. There are some benefits of this choice. Performance is one of them, and it avoids conversions in the execution of system calls.

In DSM mechanism, the data transfer cost is equal to the data transfer cost of one page. This sentence is also correct for certain conditions, especially in heterogeneous systems in which there is an operating system that uses a different memory page size other than 4 KB. In such situations and to transfer information between machines from the view of the local memory manager, a unit known as the segment is used. Each segment is assumed to be equal to some memory pages, which may be even a decimal number. The calculation of the average of the data transfer cost used is presented in Eq. 3.

$$Average\ Data\ Transfer\ Cost = \left(\frac{Size(Data\ should\ be\ transfer)}{PAGESIZE}\right) * y_3 \tag{3}$$

In Eq. 3, $y_3$ = weight coefficient to determine the importance of average data transfer cost, which must be calculated by the administrator.

## 3 Real cost of application-specific dependent parameters

The second group of parameters is due to the specifications of programs that utilize the DSM mechanism (these parameters caused by the inherent activity of DSM system and basic operations of this system).
Application-Specific Dependent Parameters:

- Page Multi-Access Overhead Cost
- Page Transfer Overhead Cost

- False Sharing Cost

Activity costs must (I) indicate the nature of activities currently running in the system and (II) should represent inherently and identity features of the system. The primary focus is on costs made by the nature of the system. Principally, this is because when an activity is executed in the system, it utilizes specific features of the system that, in turn, give meaning to the execution of the activity in the system.

Regarding such an approach, it is assumed that the occurrence of data (or function) transmission activity in the system causes activity type costs appear in the DSM production system. The very first cost made by information transmission in DSM and establishment of IPC is a cost called multi-access to DSM data.

Before entering the details of the mentioned costs, an overview about DSM system and its features should be noticed. In addition, assumptions of the DSM system should be decided for its cost extraction when there is a discussion about these details.

## 3.1 DSM operation and activity measurement units

When producing parameters causing activity costs, a unit needs to be considered for running activities in the DSM system. Software DSM systems implemented in the operating system can be thought of as extensions of the underlying virtual memory architecture.

On the other hand, considering that DSM is a kind of memories managed by HPC systems manager for IPCs, the unit of this memory must adhere to the pattern of the units used by the operating system to manage the memory [65]. Therefore, DSM access unit should utilize one of the units used by operating system to access memory. Software DSM systems also have the flexibility to organize the shared memory region in different ways. The page-based approach organizes shared memory into pages of fixed size. In contrast, the object-based approach organizes the shared memory region as an abstract space for storing shareable objects of variable sizes. Another commonly seen implementation uses a tuple space, in which the unit of sharing is a tuple [66].

The proposed model is obtained by DSM systems, which use page and segment patterns to build a structure of DSM. Although segment size is subject to variation as time passes by, yet it can cover object-based DSM mechanisms. In this paper, segment patterns are used as the measurement pattern of DSM.

## 3.2 Assumptions for calculating application-specific depends on parameters

The application-specific dependent parameters cost, firstly, indicate thinkable situations of DSM system when an activity is running in the system. Secondly, they emphasize on the basic concept of the memory model, i.e., data consistency concept. They represent the interaction of this basic concept with other elements of the production system (processes) and environmental elements of the production system (operating system and HPC system Manager). Thirdly, the mentioned situations indicate general categories of operational costs, which may occur in DSM production line system.

Selected situations to calculate activity cost covers situations of two elements of the production line, i.e., process and the data, whose consistency should be maintained. The first case occurs in a system when processes show cost-producing behaviors. The second case happens when processes break production line's sequence. The third case happens when data faces challenges in maintaining its consistency.

In this case, situations in which one of the two elements or the both produces the cost in DSM production line are considered.

This means:

(A) Several processes (worker) access a single segment.
(B) The production line is not straight, and processes, thus (workers), ought to change segment transmission stream in the production line.
(C) Processes (workers) have no information from one another in piece production.

Other possible situations for data and process are situations in which a technical defeat has occurred in the production line.

Surely, some costs may be produced in Grid and P2P systems, which cannot be explained explicitly by the mentioned costs. However, they can be mapped to the situation of the segment and processes, which were discussed in this section.

## 3.3 Model to calculate activity cost in DSM

The following relation is used to calculate the DSM cost:

$$\begin{aligned}
&DSM_{Cost_{Model}} \propto \left[DSM\ Activity\ COST \blacksquare DSM\ Operation\ COST\right]\\
&So\ that\ DSM\ Activity\ COST \propto \begin{bmatrix} Page\ Multi\ Access\ Overhead\ Cost \blacksquare \\ Page\ Transfer\ Overhead\ Cost \blacksquare \\ False\ Sharing\ Cost \end{bmatrix}\\
&and\ So\ that\ DSM\ Operation\ COST \propto \begin{bmatrix} DSM\ Create\ Cost \blacksquare \\ DSM\ Maintenance\ Cost \blacksquare \\ Average\ Data\ transfer\ cost \end{bmatrix}
\end{aligned} \tag{4}$$

The most important question that is asked about Eq. 4 is the reason of the absence of the costs, which were not mentioned. Any cost must be in operational cost group or activity cost group due to the calculation of production line cost and equivalence of production line and DSM. There is no other group for the cost of the production line and DSM system to be considered.

The activity cost consists of costs based on distributed memory and basic transferring operations. Distributed memory is similar to the other memories that consist of process and data. In data, the most important feature to be considered in DSM is maintaining compatibility. In DSM, except creation, transferring and consuming, another activity called data compatibility can be considered for data. Due to the existence of more than one process in the management scope, there is a need of using the mechanisms of data consistency establishment in DSM system [67,68]. The cost related to data consistency establishment is considered in data transferring unit cost of DSM system. Data consistency is similar to costs created by changing the transferring unit in local manager (Operating system) and HPC system manager and is part of the essence of DSM system.

It should be noticed that a rule called production rule holds for any production line. This rule talks about totality, identity, and basic features of the production line. Activity costs are raised in production line when this rule is violated.

Data consistency is considered as the primary rule in DSM production line, as in memory model.

Data consistency in DSM production line should be supported at the level of computing systems with distribution property. Therefore, violation of the production rule may be due to both an element of production line system and its distributed nature. This issue is considered as one of the innovative applications of systems concept in producing execution cost of programs since they both investigate costs caused by both fundamental elements of the system and the concept of the system.

The false lock and false sharing activity behaviors for beneficiary processes in DSM are remarkable except for creation, transferring and consuming. These two behaviors are caused by the nature of being DSM system [65,69,70]. Except false sharing and false lock for beneficiary processes of DSM and the behavior caused by consistency for existed data in DSM, any other process and data behavior that are part of producing, transferring, and consuming are operational.

However, three types of activity costs are considerable for such a system.

One of the important criteria of DSM is its transparency and hiding underlying complexities from users. However, this affects some costs calculated in the paper.

Therefore, this model is calculated for DSM mechanisms implemented at a low level. This is because DSM mechanisms implemented at the application level show less support for transparency for users. Software DSM systems implemented at the library or language level are not transparent, and developers usually have to program differently. Transparency pattern leads to a concept called hidden cost of DSM. On the other hand, as it will be mentioned further in the paper, one of the most important concepts of DSM that should be respected by this model and any other memory model is data consistency. Various models have been introduced to maintain data consistency in DSM mechanisms. These models differ in efficiency and cost. The Sequential Consistency pattern model is proposed here for the data consistency pattern.

The important point in Eq. 4 is the method for finding the relation between each of the effective parameters on the right side of the relation with the DSM real cost, for which the *dependency separation* principle is used for calculation.

According to this principle, calculating the relation of each parameter with DSM happens. The algebraic sign was found using the dependency polynomials in the case of existence of dependency between the main parameter and the other parameters. Then, the dependency and the value would be applied to such parameters by a coefficient. The sign $\blacksquare$ used among the elements in Eq. 4 represents a mathematical composition relation.

The most important point here is that Eq. 1 has been written in the form of a mathematical relation. To convert the relationship, a constant number known as the DSM constant is multiplied to the right side of the above-mentioned relation.

Based on data element and process element, two total costs are expected from activity in the Application-Specific costs related to DSM. Costs caused from being distributed and costs caused from being memory manager. The second type of costs caused the division of memory to manageable parts like segment and page by DSM manager. The costs resulting from being distributed are caused by the lack of a centralized manager of DSM. In DSM mechanism opposite to old memory mechanism, the central element does not manage DSM because physically, each part of DSM is under the management of a local operating system related to computing element.

Costs resulting from being memory manager are kinds of costs that must be paid for using the DSM mechanism in shared memory like memory management mechanisms; DSM manager element needs to divide the memory into sets of units like page and segment to manage the unit. Therefore, the first concept of activity cost in DSM results from being memory nature and instinct. The concept of memory divided into a number of units is for management. The concept of second activity cost in DSM is caused from being distributed.

### 3.4 Paging overhead cost

In order to save or retrieve some blocks located in the local memory, the local operating system converts a user access to a variable in a local memory system. In such cases, the smaller the size of the blocks, the higher will be the frequency. Bigger-sized blocks save on the frequency of accesses but may lead to internal fragmentation.

In DSM, which is naturally used to share a high volume of mostly complex data, the above problem is more critical. When a user asks for a variable, the DSM manager must either store or retrieve a big number of blocks, which in most cases are not on a single machine and are in the DSM space. Thus, with a decrease in the block sizes, the number of references to the DSM manager increases. Since such references are performed on the network, the resulting overhead is high. When block sizes are big, less memory is wasted because such systems are naturally associated with big data. Paging overhead cost is a data caused activity cost.

Paging overhead cost is related to data transmission when data is not accessed [70]. This cost is also based on activity since it would not make sense unless the IPC is done through DSM. This cost is based on data consistency principle in memory models. This cost indicates the nature of DSM model. The reason for focusing on this cost as an activity cost in DSM production system is due to realistic nature of its calculations. In theory and when investigating out-of-machine IPC mechanisms like DSM mechanism, it is usually assumed that two processes access shared data. However, in Grid and P2P systems, several processes communicate with each other (more than two) and access shared data. The mentioned cost points to data consistency maintenance when several processes access data.

One of the significant challenges of the production line is a selection of segment size. This challenge is more severe regarding production lines in which when one worker is running an activity on data, no other workers have the permission to access it. As mentioned, the main assumption of this paper about despotic rule over DSM production line is that its data consistency model is sequential consistency [71].

The complex state can be considered for a production line in which more than one worker could work on a product in a time unit. However, in such states, the activity of products does not affect each other. This indicates that the size of production unit was not selected properly. In DSM literature, this concept means that the size of segment or page is not selected logically. Selecting production line system as an equivalent system of DSM implicitly represents the use of a memory model of sequential or consistency from the strong type.

When activity performed by worker on the segment is known, the selection of segment size is easier. However, it gets harder if the worker locks the segment, i.e., no observation can be made. However, if segment size is selected larger, it may make workers wait for some time in the production line and consequently produces cost.

If it is chosen smaller, then the workers would need to lock more segments to fulfill their task; thus, the block time is increased. Also, selecting a small size in case workers' operation pattern shows they are working on large segments; another cost called sequential segment transmission between workers is introduced.

Considering an HPC system composed of $N$ machines, holding data in a distributed manner and presuming that the base size of the block is also X bits, the page overhead cost of the HPC used Eq. 5

$$\begin{aligned}
&\mathit{Paging\ Overhead\ Cost} = \mathit{Cost}\ (\mathit{Breakdown\ or\ Merge})\\
&\quad + \mathit{Cost}(\mathit{Control}) + \mathit{Cost}(\mathit{Consistency\ Keeping})\\
&\rightarrow\\
&\mathit{Paging\ Overhead\ Cost}\\
&\quad = \left[\left[\sum_{i=1}^{N} ((\mathit{Sizeof\ Sharing\ Data\ \ in\ Machine\ i}) * (\mathit{Cost\ Sharing\ Data}_{local})\right]\right.\\
&\quad * \mathit{Size\ of\ Block\ in\ MachineiX}) * \mathit{Cost\ Data\ Block} + \mathit{Cost\ Control}\\
&\quad + \mathit{CostConsistency\ Keeping} * y4
\end{aligned} \tag{5}$$

In Eq. 5, if the computing system is a closed system like Cluster, the costs of cost sharing data local and cost data block is an activity cost that fixed type, and if the computing system is a biological system like Grid and P2P, the mentioned costs must be calculated per each global activity. There is a need for the information about the nature of DSM that is used in a computing system for calculating the costs of cost sharing data and cost data block. Cost sharing data and cost data block are given in the following:

(a) Level of implementation of DSM mechanism
(b) Compatibility mechanism of data, which is used by DSM
(c) The average number of beneficiary process in shared data
(d) The system calling number for storing and achieving data
(e) The mechanism that is used in operating system for preventing from accessing the common data (like Semaphore or Monitor)

(f) The average number of required memory units for executing an instruction of beneficiary program in DSM
(g) The average probability of simultaneous accessing to existed data in DSM by two (or more than two) beneficiary processes of DSM
(h) The average number of required parallel processes for executing a global activity

The nature of effective data on costs of cost sharing data and cost data block is a statistic and follows the pattern used by the program for accessing the stored data in DSM and memory management mechanism in a computing system. Looking at the effective things in costs of cost sharing data and cost data block, if DSM manager is implemented at the kernel level of operating system, it can extract the information using existed data in data structure related to memory (and following that, DSM). Generally, in operating systems, sets of the data structure are being used for storing the data related to memory (as DSM is a memory).

In Eq. 5, $y_4$ is weight coefficient to determine the importance of block size cost in a machine. Memory is divided into $N$-bit blocks. Also, it is assumed that if a process needs to access only some part of a block, DSM mechanism locks the whole block. So the locking cost is completely related to the nature of existing data in blocks.

It is obvious that if the nature of stored data in a block has a high frequency of using computing processes, the cost of maintaining this block by memory management element will increase. On the other hand, one of the differences of DSM cost computing is in P2P systems and traditional computing systems. In traditional computing systems, as all the computing processes are related to a global activity, the nature of created computing processes is similar to each other with little different behaviors. If the high-frequency process uses a block of memory, this behavior will be repeated for a long time. In computing systems like P2P and Grid, as there is more than one global activity in the system, the computing processes will have different behaviors in the frequency of using the memory blocks.

DSM management unit should pay a cost called information maintenance. Today, operating systems handle the division of memory into blocks, and thus, the cost of data maintenance of blocks is paid by the operating system.

If DSM uses another pattern for partitioning memory other than what is used to partition main memory data, the maintenance of information status must be considered as another activity-based cost.

Another point about the cost is the use of the patterns regarding the optimization. Equation 5 shows that the cost would be reduced if the size of shared data and $X$ change. However, $X$ should be amended to become as a coefficient of the block size of each machine, and the least cost is required for its calculation.

Paging overhead cost consists of two concepts. Part of this cost caused from being memory manager is DSM management; therefore, it must have a mechanism for management's DSM. The manager of an operating system uses a pattern for managing the real memories that are constituent of DSM. DSM developmental pattern is not a centralized pattern.

### 3.5 Page transfer overhead cost (multi-access event) known as DSM consistency as maintenance cost

For using any memory pattern for inter-process interaction or communication, data consistency is considered as an important issue. Maintenance of data consistency is easier when memory unit (model) is under the control of a single operating system, which is controlled by an administrative domain constituted by several operating systems [72]. Therefore, data consistency mechanisms are considered as a significant challenge in DSM systems [69,72]. Several patterns have been introduced to maintain data consistency when several processes share, i.e., read or write, a common data. Data consistency maintenance cost is an operational cost.

Data consistency is one of the basics of memory models. Consequently, activity is expected to make use of this concept when formed in the system. Otherwise, the activity does not make use of system attributes, and thus, running activity on the system is unreasonable. Paging transfer overhead cost is a consistency caused by activity cost.

This cost is hidden until an activity runs in the system. However, this cost is entirely dependent on the efficiency of DSM system and may differ in different systems [73]. The occurrence of this event in DSM system means that, in the production line of DSM, several (worker) processes communicate with a shared segment. Modifications to the segment should not violate segment concept at any instance of time. The memory page transfer overhead does not always exist in a distributed HPC system but is only triggered by certain situations. It is not a fixed parameter and is only activated due to the occurrence of certain conditions; this parameter is called the *firelight parameter*.

This cost indicates the relation between internal systems concept with operating system's environment element (as vice representative of machine's hardware element). A coherence protocol, chosen by a consistency model, maintains memory coherency.

The problem of page transfer overhead is when data in the same data block has to be updated on various machines in the HPC simultaneously. Page transfer causes a high number of data blocks to be transferred back and forth between different machines with less appropriate action done.

Data transfers inside systems that use DSM mechanisms are costly, and the DSM manager uses the block concept for saving and retrieving data while the block size is different from different parts of DSM. If at the time ($T$), some machines in the HPC system proceed with updating or changing data of an individual block of memory, ignoring data compatibility issue, for the time being, a high volume of memory page transfer occurs in the cluster system. These transfers are called as *non-grasp transfers*. In non-grasp transfer, the size of data blocks transferred does not affect the overhead. The non-grasp transfer causes a DSM page to be continuously transferred between machines.

Data contained in the transferred memory page changes as time passes. This is to say that if $K$ machines want to change data in a memory page located in DSM (based on the data compatibility protocol used in DSM), each of $K$ machines will take the memory page from the owner machine to add to address space of itself process. The non-grasp transfer, therefore, occurs when the program that is supposed to be executed in the cluster system is divided into parts that use several shared variables. If the bigger number of shared variables among subprograms running in an HPC system produced, then the probability of non-grasp transfer occurring becomes higher. To solve this problem, running programs on the HPC system should satisfy the following condition:

$$\begin{aligned} &if\ \ S_i = Set\ \ of\ \ Variable \in\ \ Partition_i\ \ of\ a\ program\ \ then \\ &\qquad \forall i \wedge j \ni i \wedge j \in Program\ Partition \rightarrow S_i \cap S_j = \emptyset \end{aligned} \tag{6}$$

If the above condition is satisfied, it is a guarantee that non-grasp transfer never occurs in the HPC system. However, in practice, enforcing such condition for running programs in an HPC system requires making some changes to the program in a way that there are no shared variables between various parts of the program to be partitioned. The other solution is for the manager of an HPC system to find a partition for the program to be executed on the HPC. Further discussion on the first solution is outside the scope of this paper, but some compilers like Mentat [74] try to implement the first solution. The second solution is suitable for small programs and is not usable for a real program run in HPC.

Activity costs caused by the change in functionality pattern of system processes, DSM system should have a logical production approach. If for any reason, some change occurs in the production line, it would be regarded as an activity cost. In DSM system, there is a concept called multi-access to data. This notion is caused by the change in functionality pattern of processes. Non-grasp transfers cost is an operational cost caused by the change in DSM production line's pattern. The memory shared among many processes is passed over processes serially. However, time spent on moving data between processes is less than the activity time performed on the data by processes. However, an idle DSM system assumes that this situation never happens, but in real world changes from the production systems viewpoint, the changes in behaviors of processes and changes in production pattern produce this cost.

Two types of costs are created in the system when non-grasp transfers occur in DSM production line. One of them is data transmission cost, which is produced by processing, and the other one is the time cost system that waits for non-grasp transfers to complete. Non-grasp transfers have a ring-like nature, i.e., a shared data segment is turned around processes, and finally, it will be transferred to the machine start ring or next machine that needs the data. Each transfer between non-grasp ring's vertices will create a cost called non-grasp transfers. Also, the overall time spent on transferring data in the non-grasp ring will add another cost to DSM system called non-grasp completion time cost.

To calculate the non-grasp transfers cost, the frequency at which non-grasp transfers occur in DSM production line system and the time required for a non-grasp cycle to complete data transmission in the system need to be known.

Assuming (f) shows occurrence frequency of non-grasp cycle in DSM production line system, *Complete Non-grasp* variable shows completion cost for data transmission in the cycle and *HPC Uptime* is HPC system cost timing unit. Equation 7 can be used to calculate non-grasp transfer cost.

$$\begin{aligned} Non\text{-}grasp\ \ Transfer\ \ Cost = f_{M\mathrm{th}} * \big((HPC\ \ Uptime_{cost} * y5) \\ + \ (Complete\ \ Non\text{-}grasp_{cost} * y6)\big) \end{aligned} \tag{7}$$

In Eq. 7, f represents occurrence frequency of non-grasp cycles when the *M*th machine initiates data sharing. HPC system time cost is the same as HC system maintenance cost in a specific interval. An extension of DSM maintenance cost is based on time concept.

In Eq. 7, the assumption is that non-grasp activity is an atomic activity and all of it is an activity from the perspective of DSM manager. This assumption considers that the occurrence of a non-grasp activity in DSM production line is the disruption of the natural process of the production line. Therefore, DSM manager considers all needed activities as an activity for coming back of production line to a natural state.

From the perspective of DSM manager, the activities that happen during non-grasp operation do not sort activities that are in the event of DSM production line. So, DSM manager considers as only cost in the field of non-grasp transfer. The reason of this is having the atomic nature of non-grasp activities and non-grasp transfer consequently. DSM manager assumes that DSM production line must be kept active during non-grasp transfer, and here, the cost of keeping production line active is the only cost. Y5 coefficient shows the number of machines involved in the non-grasp transfer. If all the machines are members of DSM production line being involved in the non-grasp transfer, the y5 coefficient will be one.

On the other hand, it is important that a DSM production line subsystem is created during the non-grasp action. This means (a) some part of DSM manager must manage these processes and data because the set of processes are on stream on data, and (b) non-grasp action creates DSM production line in the main production line. Therefore, this cost must be calculated. (In the calculation of DSM cost, in the level of the main production line, all of the actions related to non-grasp action are considered as one action, but for the calculation of the cost of non-grasp action, a sub-production line has been assumed. The most important advantages of this method are that if sub-production line causes the creation of other sub-production line inside it, Eq. 7 can be easily calculated.)

In this situation, y6 coefficient shows the number of DSM elements involved in DSM management. If independent DSM manager is needed for managing the created sub-production lines by non-grasp action, the mentioned coefficient will be one. On the other hand, in this paper, the sub-production line pattern is used for calculating the cost of page transfer overhead. This indicates that if more than one non-grasp action is created in part of DSM production line without changing the cost calculation equation, Eq. 7 can be used and its developed state will be proposed in Eq. 8.

On the other hand, complete non-grasp cost is transfer cost in a certain interval of time. Thus, Eq. 8 can be used to calculate the non-grasp cost in machine m.

$$\text{Non-grasp Cost of Machine M} = f_{\text{M}} * ((\forall i, j \in (\mathit{UnawareCycle}: \left[\begin{array}{l} \left(\sum_{i=start\ DSM}^{Kill\ DSM} \mathit{Control\ Message}_i\right) \\ * \left[\sum_{i=Start\ DSM}^{Kill\ DSM} \left(\frac{\mathit{data\ transfer}_i}{\mathit{Page\ Size}_i}\right) * y1_i\right] \\ \left[(\mathit{Cost\ of\ Cluster\ Management\ Software\ based\ on\ DSM\ license\ per\ node} \right. \\ \left. * \mathit{Sinking\ cost\ of\ each\ node}) * y2\right] \end{array}\right] * y5$$

$$+\left[\sum\nolimits_{i=1}^{\mathrm{T}=Complete\ Unaware\ Cost}\left[\left[\left[\sum_{i=1}^{N}((Size\ of\ Sharing\ Data\ in\ Machine\ i)\right.\right.\right.\right.$$

$$\left.*\,(Cost\ Sharing\ Data_{local})\right]*\left[(Size\ of\ Block\ in\ Machinei/X))*Cost\ Data\ Block\right]$$

$$\left.\left.\left.+\,Cost\ (Control)+\ Cost(Consistency\ Keeping)\right]*y_4\right]*y6\right]$$

$$\text{Therefore, Total Unaware Transfer Cost } = \sum Unaware\ Transfer\ Cost\ Machine_M \qquad (8)$$

Non-grasp transfers may not occur in a program running on HPC systems. One may observe the formation of a non-grasp information transfer cycle upon formation of any DSM.

Non-grasp transfer cost is calculated per cycle. Since non-grasp cycles in a program are a function of DSM requesting processes' behavior, they should be calculated for each program separately. Program pattern has a significant impact on the structure of non-grasp information transfer cycles. Thus, this activity cost should be calculated while the program is running on the computing system.

Calculation of non-grasp transfer cycle cost as time passes is in violation of the nature of calculating program's cost since some costs of DSM production line are merely calculated at the program's execution time. To resolve this problem, scientific applications should be noticed that utilize HPC systems and have a unique pattern. These patterns hardly ever change but are frequently executed. This means that a scientific program designed to investigate a natural phenomenon usually investigates the pattern for different samples. Therefore, some individual numbers of executions can drive out processes' behaviors.

This is easily understandable in closed computing systems like cluster systems. In cluster systems, the cluster is only running one scientific program anytime during the life of the system. In biological systems like Grid and P2P, more than one scientific program runs in a computing system. For solving this problem, any global activity in computing systems like Grid creates a computing page, which is equivalent to cluster computing system. Therefore, in biological computing systems, executor machines of each global activity create equivalence system with the cluster computing system. This concept can be used to calculate non-grasp transfer cycle cost and cost for DSM usage.

Another important point is the dependency of non-grasp information transfer cycles to data sharing initiating machines.

A directed graph can be considered per non-grasp information transfer cycles for understanding this content. The starting point of this graph is equivalent to starter machine of this cycle. The ending point of this graph is equivalent to the machine in which the cycle is finished, and the natural process of DSM production line will continue. The finisher machine of the cycle can be equivalent to starter machine of the cycle. When directed graph equivalent to non-grasp information transfer cycles starts to be scrolled, DSM must be transferred from one node of the graph per each transfer. (This data transfer is a logic data transfer and data is not transferred in real state, but the starting address of data in the shared memory of data sharing machine is transferred.) As graph scrolling is only for directed graphs, the page transfer overhead cost changing is dependent on the machine in which directed graph is started and the relation of the computing page to DSM sub-production line.

In the worst case, if there are $N$ machines in DSM production system, $N$ non-grasp data transfer cycles are formed.

In Eq. 8, coefficients in the calculation of non-grasp transfer cycles provide the probability in the calculation of the cost of removing less important non-grasp information transfer cycles. It is done by changing the location of data to requesting processes. On the other hand, it should be noticed that non-grasp information transfer cycles constitute part of activity costs. The nature of activity costs is that against operational activities, and they can be expressed as functions of the independent time variable.

The existence of parameter $f$ indicates the frequency of data transfer cycles of machine $M$. This parameter can be regarded as an important criterion to investigate non-grasp data transfer cycles of a machine. Computing system's designer makes decisions about the lowest reasonable limit for the calculation of non-grasp cycles cost. However, machines with lower non-grasp transfer cycles are due to the occurrence of some events, which can be regarded as oscillating events and may be removed from calculations after the results are calibrated.

## 3.6 False sharing cost

The other operational cost considered in this paper indicates the situation when data consistency concept moves from traditional memory model toward distributed systems world [69,70]. These systems deny the existence of centralized structures in nature. However, this makes some challenges in data consistency model of memory [67,68]. False sharing or non-grasp sharing is a cost that indicates one of the cost generating issues in data consistency model. As stated, this cost is caused because of moving from the single user toward distributed systems

The non-grasp sharing, as in non-grasp transfer, does not occur always, and its occurrence in the system might be observed as an effective parameter in the DSM cost under certain conditions.

The non-grasp sharing at a distributed multi-computer HPC system becomes meaningful when two processes may be placed on two different machines or even on one single machine. The HPC system tends to access two pieces of data that have no relation to each other in a way that such data is placed on a single data block in DSM. These two pieces of data have nothing in common except to be located at a block of DSM. Whatever is the compatibility mechanism of DSM and without prejudice to the discussion of non-grasp sharing, both processes access the block containing the data required and tend to add them to their local memory for utilization. More precisely, any local memory manager asks the DSM manager to add the DSM block containing the data in the data transfer unit to its local process address space that requested access to the memory block. Under such conditions, there is a data block that has been shared between two processes, and each of them has subjected the continuity of its work on releasing the DSM block. Under such conditions, an error might have occurred in the distributed multi-computer HPC system.

For identification of non-grasp sharing, like identification of dead-end occurrence in operating systems, the following two policies pertain to the discussion of DSM management in general:

1. DSM manager applies a series of policies such as frequent checking and inspection before the execution of non-grasp sharing process inside the system. It should be noted that such action is quite costly on the given scales.
2. DSM manager does not consider the occurrence of non-grasp sharing. Under such conditions, processes involved in non-grasp sharing will destroy themselves after finishing the dedicated execution time. In fact, in this method, the DSM manager does not involve itself in the matter of non-grasp sharing at all and allows the processes to exit the system upon finishing their execution time solving the problem.

Many DSM systems use the second method.

The second important subject in the field of non-grasp sharing is the dependency of sharing on the block size. It is practically proved that in an HPC system with the DSM programming model and mechanism, the bigger the level of the logical blocks generated by the DSM, the higher is the probability of occurrence of non-grasp sharing [75]. This is valid particularly in HPC systems that perform program distribution at the subprogram level. It also frequently occurs in HPC systems in which the program execution has a large number of variables and distributes the program at the loop level.

The important issue in the field of non-grasp sharing is the extent to which parameters are related when sharing occurs. A part of this issue is associated with the method of grain that may be approached via either the user or the compiler. The other part of this response is associated with the HPC definition. In case the HPC acts in a special way, in which only one task is done at any moment, then the level of occurrence of the non-grasp sharing is significantly reduced. In case the HPC is allowed to execute more than one task at any moment like Grid and P2P, the rate of non-grasp sharing occurrence is increased along with increases in the number of tasks.

Another factor that has a direct effect on the occurrence/non-occurrence of the non-grasp sharing is the issue of blocks in the DSM, which is considered similarly from two viewpoints. First, the probability of the existence of two independent variables in a block is considered based on pigeonhole principle.

The second issue is the non-grasp sharing dependency on the size of the block in the DSM. The purpose of the inspection is to know the probability of two variables that cause non-grasp dependency to be located in the same memory block. Task $A$ is assumed to be executed on $K$ machines, which have also been divided into $n$ sub-tasks. Thus, there is a $K$-distributed memory. Please note that the expression is division and not separation.

The relevant comparison to be made in this regard is between the values of $n$ and $K$. The problem may easily be converted into one of the set algebra classical problems. There are $n$ pigeons and $K$-holes according to the pigeon hole principle; in case $n > k$, there is a hole in which there is more than one pigeon so that the non-grasp sharing may occur.

This problem is illustrated in the HPC with $n > K$. The reduction and absurdum for this problem are used. Two states may be considered for reduction and absurdum.

In the first state, $n < K$. So practically, there are some machines that are idle. Idle machines are in conflict with the philosophy of creation of HPC systems that aims to reach a high processing capability. It is equal to have the programmer or compiler that is not presenting a proper partitioning on the concept of HPC systems, causing some machines to become idle in the HPC with no assigned tasks. Thus, the occurrence of such conditions in HPC systems is improbable. In the second state, $n = K$ and two sub-states are found. In fact, on the existence or non-existence of dependency, the state is converted into such two sub-states. The first one frequently occurs in "$n$" programs that are not dependent on each other. In fact, in such state, a subprogram is under execution on a machine and uses a piece of data during its execution that is located in the block associated with the other subprogram. This state is general, and its occurrence can be seen in most of the programs executing on HPC systems

In such state and at a certain time, the process "$i$" requests the block associated with process "$j$" to be added to its address space to continue its operations. Process "$j$" may also be working with its data in such block, so a non-grasp dependency is created in practice.

There may occur another certain state when $n = k$, in which no non-grasp dependency occurs at all. Such state hardly occurs; however, partitioning of the main program into subprograms occurs in a way that subprograms are not dependent on each other. In the case where the main program is partitioned into subprograms such that no two programs need to exchange information and the number of subprograms is equal to the number of machines, it may be said that non-grasp dependency never occurs.

Non-grasp sharing cost is caused because DSM production system is a distributed production system in which central machines are avoided. Taking a look at the nature of what causes this cost, it is obvious that this cost is due to effects of identity issues of a distributed system using DSM in HPC systems.

Non-grasp sharing cost in HPC systems like clusters is lower in comparison with biological computing systems like Grid and P2P. In a closed computing system, the system manager can gather the exact information about the program and constituent subprograms. On the other hand, in such computing systems, system manager has exact information about the machines that are the members of a computing system and has their abilities. These two subjects cause the manager to calculate the non-grasp sharing cost easily.

In these systems, there is no possibility of $n > k$ state because the manager considers this state as loading disruption because of having exact information about computing elements state. And also with using the mechanisms of migration and based on loading, the manager tries to change that state to $n = k$ or $n < k$.

If loading element can do the open activity of loading based on subprograms, non-grasp sharing cost will decrease in the system. In biological computing systems, the manager does not have exact information about computing system and its elements. Therefore, according to pigeonhole principle, the probability of non-grasp sharing occurrence increases. Due to this, in biological computing systems, especially the systems in which there are computing elements with a high frequency that are increasing or decreasing in number, the non-grasp sharing cost is increasing. In such computing systems, there is the probability of occurrence of $n > k$ state because of adding of computing machines. This causes the system manager to call the loading units and process migration mechanism with higher frequency. Calling the loading element and process migration in both biological and closed computing systems causes the disruption of the natural process of DSM production line, and it causes the non-grasp transfer cost except non-grasp sharing cost.

False sharing cost, as other activity costs discussed in the paper results from maintenance of the basic rule in DSM production line system, i.e., data consistency. Practically, this is because two irrelevant data existed in a block and were accessed by two irrelevant processes. In this case, DSM manager sends one of the processes to the blocked state to hold the basic principle of the production line. From a process standpoint and if it knows its status and status of its required data executively entering blocked state are nonsense.

The mentioned cost would appear in the real world and production lines as well. However, its appearance is highly dependent on the size of an existing piece of the production line. If the piece is large and contains several sub-pieces such that some task must be performed on each piece, as a worker locks piece $i$th to work on it, all other workers will be workless. However, it should be noticed, in nature, this blocking differs with the blocking in which two processes intend to work on a shared data and production line's basic rule prevents one of them. In this case, if the size of the piece was well appointed, it could prevent executive blockage of processes.

False sharing is a time activity cost. In DSM production line system, this cost indicates the cost exerted on the system due to blockage of a process. To calculate cost produced by false sharing, two issues must be known. First, what is occurrence probability of false sharing in HPC systems? Second, what is the frequency of false sharing occurrence regarding block size in HPC system? Moreover, finally, what cost should be paid in an HPC system for each occurrence of non-grasp dependencies? There are different methods to investigate occurrence probability of non-grasp dependencies in DSM production line system.

As mentioned in this section, the possibilities driven by pigeonhole principle can be used to decide about the status of the occurrence probability of non-grasp transparency. In an HPC system, if $g$ shows frequency of the situation in which $n > k$ for a certain time interval, then '$g$' indicates occurrence probability of non-grasp dependency. The cost of non-grasp dependency for a DSM production line system is how much time a process keeps up the shared memory distributed between it and another process while the process does not use DSM. In other words,

Non-grasp sharing

$$= g_{M} * \left[ \left[ \begin{array}{l} \left(\sum_{i=start\ DSM}^{Kill\ DSM} Control\ Message_i\right) \\ * \left[\sum_{i=Start\ DSM}^{Kill\ DSM} \left(\frac{data\ transfer_i}{Page\ Size_i}\right) * y1_i\right] \\ \left[(Cost\ of\ Cluster\ Management\ Software\ based\ on\ DSM\ license\ per\ node * \right. \\ \left. Sinking\ cost\ of\ each\ node) * y2\right] \end{array} \right]_{interval_t}^{Process_{i,j}} * y8 \right. \\ \left. + Waiting\ Cost_j * y7 + Running_i\ Cost \right] \tag{9}$$

In Eq. 9, assuming occurrence probability of non-grasp dependency for process $j$ in a machine that is a member of DSM production line system is equal to $g$, this cost states process $j$ keeps the data it has shared with another machine in which process $i$ exists plus waiting time of process $j$.

In the calculation of Eq. 9, like Eq. 7 and its development mode, in Eq. 8, DSM manager considers non-grasp sharing as an atomic activity. This is like Eq. 9, where non-grasp sharing is considered as management element of activity that the natural process of DSM production line wrecked. From the perspective of DSM manager, the only cost of DSM manager when such a thing happens is DSM uptime. If two processes $i$ and $j$ in $t = 0$ (time) want to access the common area in DSM, the DSM manager decides about non-grasp sharing following its policies that process $j$ gets suspended. If there is a need of $t$ time unit for executing the process $i$, from the perspective of DSM manager, the maintaining cost of DSM between $i$ and $j$ processes in $t$ time must be paid. The nature of non-grasp sharing cost is highly dependent on the process of execution of the program in a computing system. This means that the method of communication between machines that are the members of a computing system, the ability of each machine in executing the program and does computing element which performs subdivisions of activity required to be synchronized with each other or not? They are effective on the cost of non-grasp sharing. On the other hand, it should be noticed that despite all activity costs mentioned in the paper, the calculation of the cost of non-grasp dependency is highly dependent on time. In experiments performed to investigate non-grasp dependency status, it was recognized that some of the dependencies appear in a majority of experiments and some others are different from experiment to experiment. In fact, they are a function of completion time of the activity that involves the shared data.

This means some non-grasp sharing activities might occur in the specified scientific program and some non-grasp sharing activities might not occur in other program running the scientific program. This defines the y8 coefficient. Y8 coefficient shows the occurrence of non-grasp sharing activity related to processing $j$ in comparison with total activities related to DSM by process $j$. In the worst state, this coefficient is equal to one, and this means that every time process $j$ is using DSM, non-grasp sharing activity occurs. The information related to the y8 coefficient for each process $j$ from the data structure of process state and data structure related to memory accessing is extractable by process $j$.

This cost can be calculated by data structure related to an operating system that maintains the running error of process and the data structure related to IPC of out machine process. This cost is per system call.

Another part of non-grasp sharing is related to waiting process $j$ for ending the mode of non-grasp sharing. The cost of waiting process $j$ is all system calls, which are the reason for suspension of process $j$ in performing the global activity of which process $j$ is a part. So, this cost is covering the cost of system call performing that the existed other processes in a computing system that is for communicating with $j$ process. These costs can be calculated by data structures related to an operating system that maintains process execution error and the data structure related to processes communications of out of the machine.

It should be noticed that waiting time of process $j$ might be very complicated since it contains communications of process $j$ with a set of processes waiting for process $j$ to finish its task with DSM so that they can resume their activities. In this paper, for simplicity, it is assumed that wait cost of process $j$ is calculated based on central processor's cycles and the importance of process $j$, which is given by $y7$.

The y7 coefficient in Eq. 9 shows the importance of coefficient of the process in global activity. If all activities related to process $j$ were global activities in runtime, this coefficient would be equivalent to one. For calculation of this coefficient, the ratio of the numbers of global activities on all activities of process $j$ is considered.

Similar to Eq. 7, during the running of non-grasp sharing activity, non-grasp sharing activity is the only existed activity at the level of the system from the perspective of a DSM manager. A DSM sub-production line can be considered if there is a computing element in which process $i$ is running and there are other computing elements in communication with process $i$. This system contains another computing element in which process $j$ is running. In this DSM sub-production line, DSM manager can be defined. In this sub-production line, the existed process $i$ has activity on DSM. This process performs a set of system calls that were defined for performing on DSM. The methods of timing cost can be used for calculating the cost of each system call such that a cost is considered for each time unit and the cost of the system call is equivalent to the cost of required time for running the system call.

In a computing system, especially biological computing systems, during the lifetime of DSM manager, probably more than one non-grasp sharing activity can occur. Equation 9 can be used for calculating the cost of each activity.

## 3.7 Specifying the relation between effective parameters

To find the sign of parameters in the necessary relation, the indexes to its right side of Eq. 4 are classified into two groups: indexes with either a weak or strong relationship between them and indexes that have no dependency on each other.

The nature of manufacturing system is such that for extracting the total cost, the existing sub-costs in the system are being added to each other. Adding the costs with each other is the normal form of communication between existing sub-costs in a manufacturing system. The reason that the operator between composing elements is added in manufacturing systems is due to the nature of cost. In manufacturing systems, the cost composing elements are extracted in a way to have the lowest amount of effectiveness with other cost composing factors. This causes each cost to have the lowest amount of dependency to other costs.

This is true about DSM system. The default operator between composing elements of DSM using cost is the added element, except the situation in which the dependency between composing elements of two costs with each other is very strong. So, in the calculation of the sign between composing elements and the cost of using DSM needs, there is a need to have enough information about costs composing elements which cause the costs to have communication with each other.

In the DSM system, the correlation between a constituent parameter of operational sub-costs is weak. Therefore, the operator that makes the connection between sub-costs with each other is equal to add operator. So, in [63], based on the fact that the only thing that makes the relation between operational costs is information transfer element, the relation between constituent costs and operational cost are considered as sum.

In the DSM system, some special modes can be considered in which the operational costs are dependent on compatibility and the size of the block. In this situation, the costs relating to these two contents are as a communicator between the composing elements of operational costs with each other. The communication between the composing costs of operational cost occurs through these two costs in which the communication between composing elements of operational cost is moderate dependence.

So, in constituent costs of operational cost, when the content of data block size is proposed in any form like block size or data size, the correlations of parameters between constituent elements of activity costs contents should be checked again. In this paper, based on the content that the size of data is a content which has been extracted out of DSM and has come from the process (labors) in DSM composing systems, the mentioned correlation has not been calculated. It is clear that if the content of data size is considered as a variable in DSM system in the calculation of operational cost, there will be a need to calculate the correlation of composing costs of activity cost based on data size.

Using the content of element that makes the connection between composing elements of the cost is more complex with respect to activity costs. Surveying nature of activity costs shows that the factor that creates the correlation between constituent sub-costs of activity cost is not only dependent on the data block size. The operator between each sub-cost shows the correlation of these sub-costs with each other.

In this part, it is supposed that based on the total nature of production line and its composing sub-costs, the add operator shows that sub-costs have a weak correlation with each other. In this situation, the distribution of the sub-cost elements is normal. The reason is clear. In a normal distribution, the freedom degree of variables from each other is unlimited. Therefore, their correlation is weak. On the other hand, based on the experiments about sub-costs and their correlation effects on each other, it is obvious that in time of strong correlation between related data and two sub-costs, the effect of two variables on each other can be described by * operator, and it also shows that the relation between two variables with each other is nonlinear.

In this paper, the sign extraction method is used based on the data used for specifying the relation between constituent costs of activity cost. The total cost of using DSM in a system can be specified based on the given equation in the second part of the paper, operational costs, the proposed equation in parts 3.3–3.5, and activity costs. By considering DSM as IPC on a specified distributed computing system, the sign between constituent elements of the activity cost is set. In this paper, the cost of using DSM in 300 times has been surveyed. The experiment environment in part 4 has been described. In these 300 experiments, it is supposed that non-grasp transfer (NGT) and non-grasp dependency (NGD) costs are actually one cost and activity cost relating to DSM include non-grasp (NG) and paging overhead cost. The surveyed data in these 300 times experiments are the amount of each of NG cost and paging overhead cost and total activity cost of using DSM. In these experiments, each of these 300 experiments shows how NG cost and Paging overhead cost influence total activity cost and their communication with each other in order to create total activity cost accordingly. The results of these experiments have shown that these two costs have a strong linear correlation with each other.

The first activity cost introduced in the paper is paging overhead cost. This cost is completely dependent on the size of the data segment, and since this is an external factor, it has a linear correlation with operational costs.

Effective variables on this cost keep the DSM system active and wait for the cost of processes while paging overhead cost results from data and its size. The only connection between the NG and paging overhead cost is waiting time of the process. However, waiting in paging overhead cost is different from that in non-grasp dependency. Consideration of data size as an external factor for the paging overhead cost reveals a weak relation with other costs.

Two non-grasp costs mentioned in the paper have a strong correlation because the occurrence of one in the system increases the occurrence probability of the other.

Performing '$n$' observations on DSM production system to extract relation between non-grasp transfer, dependency and assuming that number of estimators which calculate effects between the two real value generating sub-spaces are equal to some members of Estimator set, then Estimator set is a set which represents all thinkable states for math signs. Math signs represent the effects of non-grasp transfer and dependency on each other. Equation 10 can be used to describe this state.

$$\begin{aligned}&\forall i \wedge j \; so\ that \; i \wedge j \in \{\text{non-grasp Dependency } (NGD)\,, \text{non-grasp Transfer } (NGT)\}\\&NGD, NGT \in DSM \; can\ define \; \beta_i \circledcirc \beta_j \; also \; (\circledcirc \in Rater)\end{aligned} \tag{10}$$

Equation 10 tries to show that per each $i$ and $j$, which can be member of $\{\text{non-grasp } \mathit{Dependency}\ (NGD)\,, \text{non-grasp } \mathit{Transfer}\ (NGT)\}$ and NGT and NGD be members of surveyed DSM system, $\circledcirc$ evaluator can be used for describing costs relating to $i$ and $j$. Equation 10 introduces the fact that with a evaluator like $\circledcirc$ and using the data relating to NGD and NGT variables, the describer operator of relation between two variables can be introduced.

Regarding cost-producing nature of non-grasp transfer and dependency and conducted experiments, Estimator $= \{+, *\}$. $+$ operator represents weak dependency and is driven by cost nature of production systems. On the other hand, the $*$ operator indicates strong effectiveness and existence of nonlinear relation. Now, based on a concept called real correlation, an estimator is driven out to determine non-grasp transfer and dependency. This estimator should provide an accurate interpretation of effects of non-grasp transfer and dependency cost functions.

Considering $R_k$ as an estimator for measuring transfer correlation and non-grasp dependency and assuming $R_k \in Rater\,[NGD - NGT]\,$, $O_{\beta_i}$ as observations of NGT sub-space and $\overrightarrow{O}$ as average of observation set $O_{NGT}$, then deviation of observations centering on the average, i.e., $\overrightarrow{O}$ is:

$$\sum (o_i - \underset{O}{\rightarrow})^2 | o_i \in o_{\text{NGT}} \tag{11}$$

On the other hand, $\overset{\frown}{O}$ represents regression line, which can be assumed for $O_{NGT}$ set. According to this fact, deviation of observations (members of $O_{NGT}$ set) that can satisfy regression line is obtained by Eq. 12.

$$\sum (o_i - \overset{\frown}{o})^2 | o_i \in o_{\mathrm{NGT}} \tag{12}$$

In this method, NGT is as affected sub-space:

$$(\mathrm{NGT} - \mathrm{NGD})^2 = 1 - \left(\sum (o_i - \overset{\frown}{o})^2\right) \Big/ \left(\sum (o_i - \underset{O}{\rightarrow})^2\right) | o_i \in o_{\mathrm{NGT}} \tag{13}$$

Zero results in this Eq. 13 show that $O_i NGT$ sub-space completely produces under estimator $K$ effects on NGT. It should be noticed that correlation is always a number between zero and one. One indicates under estimator $R_k$ effects on *NGT* are completely produced by *NGD*, and zero indicates under estimator effects on NGT are not completely produced by *NGD*. However, the correlation will be zero in case $\sum (O_i - \overset{\frown}{O})^2$ equals $\sum (O_i - \underset{O}{\rightarrow})^2$. This shows that under estimator $R_k$, the effect of *NGT* from *NGD* sub-space can never be investigated. In DSM production system, the estimator set is described as below based on observations provided by running MM5 application [26], which is rewritten based on a DSM mechanism:

$$Rater_{RA} = \{*, +\}$$

Correlation theory is used to determine which operators, + or *, can better express the effect of two NGT variables on NGD variable. To recognize which of the two above estimators can better state the correlation between NGD and NGT and more accurately determine the effect of NGT variable on NGD, a set of data needs to be produced.

Therefore, the costs of non-grasp transfer and dependency for a certain DSM production system are calculated for a certain number of times for * estimator. Then, the table is investigated for the + estimator. It means that non-grasp transfer and dependency are calculated while estimator is +. The results are provided in a table similar to Table 1.

Table one shows the effectiveness of *NGT* on *NGD* when estimator is * or +. Upon completion of the table for 300 times, the correlation coefficient can be used to determine estimator.

**Table 1** Values of *NGT* and *NGD* for * and + estimators

| + estimator | | * estimator | |
|---|---|---|---|
| *NGT* | *NGD* | *NGT* | *NGD* |
| *NGT1* | *NGD1* | *NGT2* | *NGD2* |

Using * as an estimator, the correlation between *NGT* and *NGD* is 0.0974 while if the estimator is +, it will be 0.0862. Therefore, the accurate estimator that can accurately describe effects of *NGT* on *NGD* is *.

It can be proved intuitively. In non-grasp dependency and transfer costs that form a data block, non-grasp cycles are formed with a higher probability of release of the block. Therefore, violation of the idle model of DSM production line can increase the probability of involvement of the data block needed by other processes, such that two processes may not communicate, but one of the processes is in the non-grasp cycle. Consequently, the cost of the process would increase. This fact holds inversely as well, i.e., a cycle may be created merely since a process has a data needed by another process.

## 3.8 Calculating the constant coefficient of HPC

As mentioned previously, $K$ is called the HPC system coefficient or HPC constant. The $K$ HPC constant is defined based on the HPC capacity. $K$ is a value that causes the relation between the DSM cost and the effective parameters for it to be converted from a relation to equality and equation.

In this part, relation means Eq. 4.

An HPC coefficient is a number that has been obtained by experiments. To find the HPC constant, different parameters are considered, as shown below, to calculate HPC constant.

A. The level of calculations at each unit of time
B. The average number of messages required to set up the HPC system based on the DSM mechanism
C. The average number of distributed memories
D. The average number of blocks associated with the establishment of DSM
E. Average frequency of request to DSM
F. Average number of instructions related to IPC out machine
G. Average number of system calls to memory activity except for DSM
H. Average time of network communication (convert to cost based on time cost)
I. Average cost for network communication creation (based on time cost)
J. Average number of control instructions used by DSM (convert to cost based on cost system call)
K. Average number of errors occurred during DSM management process (convert to cost based on cost system call)

It was practically shown that none of those above parameters were the missing link in such a relation to be converted to an equation. From a physical point of view, none of those above elements may keep the dimension of the obtained relation.

Based on achievements, it was understood that all the items existing in the right side of the corresponding relation are related to an item known as the *HPC size*. The level of non-grasp sharing, non-grasp transfer and the costs of establishment and maintenance of the HPC are the main parameters that directly relate to HPC size.

**Table 2** *K* coefficient values based on the HPC size

| *K* value | Size of HPC |
|---|---|
| 0.2 | $> 1$ and $< 100$ |
| 0.5 | $> 100$ and $< 1000$ |
| 0.7 | $> 1000$ and $< 10{,}000$ |
| 0.9 | $> 10{,}000$ |

Nature of production line causes this content. All parameters of production line costs are dependent on production line implicitly. This is true about activity parameters and operational parameters.

None of the given equations in this paper says anything about the size of the environment and the size of the HPC system. It is, thus, possible that the constant HPC parameter is complementary to the discussion of parameters affecting the memory size criteria from the view of DSM manager.

Considering all the concepts mentioned above in practice, the HPC constant is a concept that first specifies a reality in the HPC systems. Second, it may be able to protect the given relation dimension, and most important of all, it may change the status of the relation (Eq. 1) mentioned above and convert it to an equation. Finding such matter and after ensuring that the parameter listed above is considered as an effective parameter in the HPC, *K* coefficient must be recognized by interpolation in continue.

If the relation between the parameters equal to *A* be considered, then there is Eq. 14.

$$Cost\ \ of\ \ DSM = K * A\ \ and\ \ K = Cost\ \ of\ \ DSM/An \tag{14}$$

To calculate *K*, a series of DSM costs are also calculated, which there are their values using the equation and their place in the corresponding relation, and some values for *K is found*. HPC system has been described in a previous study [51].

As expected, another *K* can be obtained for all HPC systems. *K* is practically different for HPC systems with different sizes. After finding the various points, in which the level of the right and left sides of the Eq. 4 were known, such relation required the assignment of different values to *K* to be converted into an equation.

Table 2 shows the numbers derived from tests using the extension principle. The maximum size of the HPC values could not be completed in Table 2 due to resource constraints at our disposal.

Table 2 has been created based on interpolation pattern. In computing system [51], the number of machines, which are members of the computing system based on a hundred coefficient, was increased and with computing the real cost and achieved cost from Eq. 1. Therefore, coefficient *K* follows Table 2 with 97% coefficient. It is important that when the number of machines that is a constituent of computing system was between 100 and 1000, different numbers gained for *k* that with experiment 97% of this numbers are in the limit of 0.5. The real cost in the system [51] is calculated based on time cost pattern and system call cost. In the calculation of coefficient *K*, the focus is based on the size of HPC system; this content implicitly covers the condition of IPC with the owner of distributed common memory and two processes that are using the common data in DSM level. On the other hand, in the calculation of coefficient $K$, effective environmental and systematic factors on DSM have been considered.

## 4 Case study

To evaluate the proposed approach for calculation of DSM cost, the P2P system with the management framework has been followed as described in a previous study [51]. In this P2P system, there are different operational regions including HPC regions. In these areas, both the IPC mechanisms are used, i.e., DSM and message passing.

DSM mechanism patterns can be converted to message-passing patterns and vice versa. A two-level out-of-machine IPC structure is used to reach higher computing performance or less execution cost. This probability is achieved through a concept called analysis of goodness cost.

The main model and the mentioned conceptual of distributed P2P system as described by a previous study [52] is based on supply and demand pattern. Therefore, there are good facilities for analysis of supply cost than demand cost. DSM is used as a place for supply and demand in distributed P2P system. When a process wants to access data, creation cost, management, and maintenance cost of DSM as part of supply cost and in answering time to request, DSM is proposed as part of the supply cost.

In this system, there is a unit called Communication Manager that is responsible for changing communication mechanisms. This unit makes decisions in the direction to reach high performance and reduce execution cost of the application program. It should be noted that when IPC pattern in the system is same as the communication pattern used in the application program, Communication Manager is activated in all machines involved in the execution of the application program. This unit handles pattern conversion through initialization of DSM and message data structures.

Communication Manager uses a pattern for goodness cost analysis that applies a concept called balance point as the fair point for the cost in DSM or message mechanisms. The most important property of this point is the determination of acceptable threshold for costs of DSM and message mechanisms. Communication Manager can determine cost function for using message mechanism based on message production system concept, in the same way as it is discussed for DSM mechanism in the present paper. Accordingly, it can calculate message production system cost function.

The main goal of Communication Manager is to analyze balance point of both DSM and message production systems. For this, it defines a concept called regional payment that completely depends on the identity of the region and is a function of two concepts. The first concept is a cost for using member machines of a region system-wide, and the second concept is the acceptable time for execution of application program on computing region.

Using member machines of a computing region in distributed systems is a concept that points to the fact that due to the geographical and administrative expansion of these systems, many of member machines of the computing region may not be present in the administrative domain of the executing machine that requires computing power.

Communication Manager converts these costs based on a unit that calculates the cost of DSM or message systems. On the other hand, the execution time of an application program that requires high performance can be calculated and estimated.

The simplest way is based on a set of environmental parameters. Communication Manager considers a cost unit for a time unit. The sum of these costs is considered as total sufficient time cost for executing the application program. The two mentioned costs are considered as payment from the system point of view. These two costs are because the user of computing regions of the distributed P2P system should pay to use the mentioned computing system.

It should be noticed that these two costs are considered as negative costs for Communication Manager, who considers these costs as its inputs. In addition, it is assumed that the elements using computing region should pay them as payment of Communication Manager.

Communication manager can analyze balance point in a more detailed manner. Analysis of this position determines how costs of the production system including either DSM or message production system, in addition to other costs of the application program, should be in order for the total cost of the system to be equal to the total payment.

Total cost from this unit point of view includes both fixed and variable costs. Fixed costs include costs that do not vary with the usage of existing machines in a region, and variable costs include costs that vary. Thus, they are presented as follows:

- $Q$ represents time or computing, and processing power is given to a requesting machine.
- $P$ cost of each time unit or computing and processing power unit.
- $F$ total fixed cost per execution of application program.
- $V$ variable cost per computing and processing power or time unit.

It should be noticed that Communication Manager calculates total costs and payments to the system based on calculations of the application program production line system. Total payment and cost are equal to $(Q * P)$ and $F + (Q * V)$, respectively. Regarding this issue, the balance point is reached when the following Eq. 15 holds as follows:

$$Q * P = F * (Q * V) \tag{15}$$

It should be noticed that code segment of an application program, which requires computing and processing power except parts that perform global operations is considered as a fixed cost. It is assumed in the paper that only IPC instructions perform global operations, and thus, all instructions except DSM/message are considered as fixed costs.

Assuming that system is investigated when the P2P system resides instability, and consequently, regions are fixed, and the payment is also fixed. In this case, in balance point equation, only IPC-related instructions are considered as variable costs. Therefore, in balance point equation, DSM or message-related costs are discussed as unknown part of the equation.

The equation is calculated for DSM production system. As mentioned earlier, a cost in DSM production system is either operational or activity. Operational costs are not a function of time. On the other hand, many parameters in operational costs discussed in the equation for DSM operational costs are not a function of machines involved in the execution of DSM based application program and are considered for a DSM production system.

Although some parameters including the amount of transferred data per usage of DSM and cost of DSM management software license are related to some machines in the system, these parameters should vary based on the level of involved machines in the DSM program execution.

For simplicity of calculations, it is assumed that the size of DSM production system, while active in the region, is a fixed and constant number. Although the proposed equation for cost and the mentioned method can be generalized for imbalance eras, it is merely necessary to divide variable parameters, which are based on some sophisticated machines in solving the problem from parameters that are not affected by the numbers in machines.

These parameters should be considered as part of parameters producing variable cost. In the mentioned distributed P2P system, DSM mechanism is implemented at the kernel level and is the concept of extending data structures used in out-of-cluster IPC mechanisms and memory management for managing DSM based out-of-HPC IPCs. Both the above-mentioned mechanisms make use of local operating system units for DSM management.

It is assumed that the operating system used in member machines in the region is Linux, with page size taken as 4096 with a good approximation. The software used for DSM cost calculation is matrix multiplication. Having another look at the proposed equation, *PAGESIZE*, *Number of DM*, *Average Cost of each unit*, *Number of Nodes*, *DSM license per node* and *Setting up cost* are considered as fixed costs and all other parameters are variable costs, which are dependent on execution.

However, in experiments of this part, the mentioned costs are considered as fixed costs. But if there would be more than one global activity in a computing system in the described P2P system, the mentioned costs can be calculated as per each global activity separately. This separate calculation will not do any damage in the experiment.

The first variable is *Size of Data should be transferred.* An experiment was conducted with matrix multiplication for 50-time units on 10 machines. During the experiment, 2–10 machines may be used for execution. Figure 2 shows the total amount of data transferred throughout DSM system per time unit.

Investigating the conducted experiment, one can notice data transmission during experiment execution is a function of the followings: (a) number of machines associated with DSM production system; and (b) type of machine, computing, and processing power of the machine, and the pattern for converting sequential matrix multiplication to parallel matrix multiplication that plays an important role in the amount of transferred data in DSM production system.

What is of high priority for cost calculation of DSM production system is the production of information transmission pattern based on time. To extract this relation, firstly, unrelated data should be removed, and secondly, the relation between existing information should be driven out. Regression is considered as a solution to both requirements. Finally, the equation $Data\ Transfer\ Size = -0.251t^2 + 11.995t + 1202.9$ is produced.

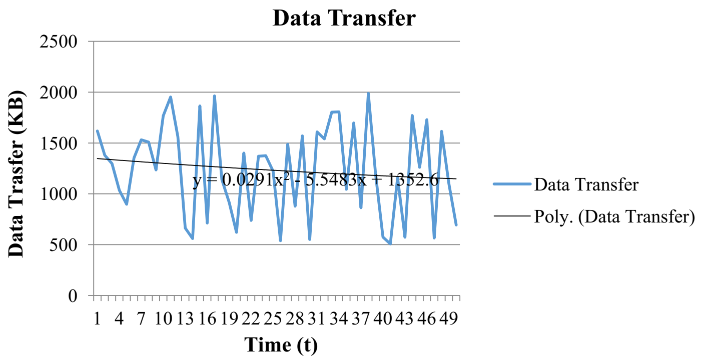

**Fig. 2** The amount of transferred data in DSM production system on time

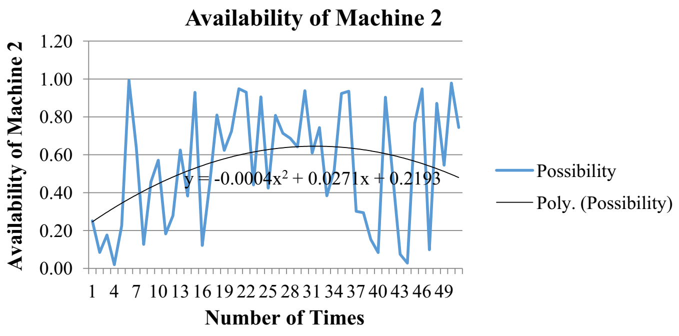

**Fig. 3** Availability of machine 2 in the matrix multiplication DSM production system

Repeating the experiment shown in Fig. 2, one can interpret that in DSM production system relational pattern introduced between the amount of transferred data in DSM and time is repetitive. The difference between various experiments is in the intercept of the function that states the relation between the amount of transferred data in DSM and time. The main reason for the difference in intercepts of functions is due to the distribution pattern and distribution method of matrix multiplication program.

Another time variable is *Average Run Cost*. This is a dependent variable with some independent variables because costs like consumed energy, the cost of operating system and cost of system and management software running on the local machine are considered as costs of this variable.

In the evaluation, without discussing details of sub-costs, it is assumed that a total execution cost in machine $i$th is equal to $Cost_{il}$. However, *Average Run Cost* in computing and processing system is dependent on activation of DSM. Matrix multiplication experiment is repeated with the mentioned conditions. Figure 3 shows the probability of the existence of a machine in DSM production system.

Figure 3 shows existence probability of machine 2 for different executions of matrix multiplication program. However, existence probability of machine 2 is low in some distributions of this program, and in some others, the probability is a number close to one. In this experiment, the lowest and highest limits are considered as 0.3 and 1, respectively. This means available machines in this range are in DSM production system.

To remove undesired data and reduce the effect of experiment's conditions and oscillations caused by them, regression is used. The equation obtained for the probability of the existence of machine 2 in DSM production system based on some repetitions of execution of the program is in the form $Number\ of\ Times = 0.0001t^2 - 0.0095t + 0.6242$. Repeating experiments for a certain number of times and the difference in goodness equations is in intercepts. Differences in intercepts of DSM operation initiating machine state the existence or non-existence of machine number 2 in DSM production system. Finally, for any machine like machine number, the proposed equation can exist for calculation of DSM cost as follows:

$$\text{Average Run Cost} = 0.0001t^2 - 0.0095t + 0.6242 * Cost_{i2} \tag{16}$$

Another time variable is *Size of Control Message*. This variable has a direct relation to some transferred messages sent throughout DSM production system to control production line.

One can calculate the number of cost units for the cost of transmission of a single unit of data. This cost is a complex cost, which includes the cost of network establishment, network protocol and required energy for the network. By nature, both *The size of Control Message* and *Data Transfer Size* variable are the same. However, they differ firstly in size, in which *Size of Control Message* is much smaller with respect to the amount of transferred data by memory units. Secondly, this variable happens when a specific event occurs in the system or a certain amount of time is passed.

Matrix multiplication experiment is repeated to extract the relation between *Size of Control Message* with time. The experiment is performed using both DSM and message mechanisms.

Figure 4 represents some occurrences of control messages in DSM and message production systems for 50 times experimental repeats.

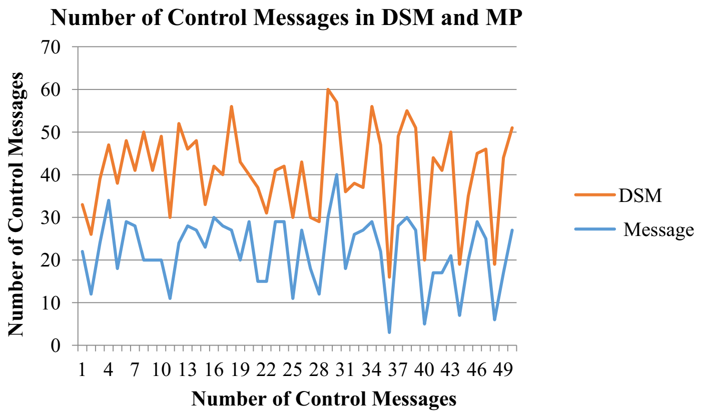

**Fig. 4** Number of occurrences of control messages related to DSM and message production (MP)

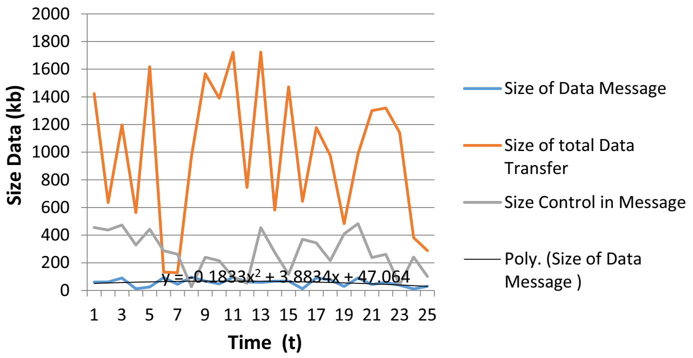

**Fig. 5** Total data transferred and amount of control data in DSM and message production systems

As shown in Fig. 4, the amount of control data that is required by DSM mechanism is larger with respect to message mechanism due to abstraction provided by DSM mechanism.

This mechanism needs larger amount of data to provide users with abstraction and control data transmission in the system. The Communication Manager can use the relative number of occurrences of control messages as a useful tool to recognize the importance of this type of messages in DSM production system.

In Fig. 5, some control messages along total data transferred during 25-time units of the experiment are presented.

As shown in Fig. 5, except three plots, the total amount of data transferred in DSM production system is 3–5 times bigger with respect to control data used in this production system. This can represent the importance of coefficient of control messages in DSM production system.

As shown in Figs. 4 and 5, in some repetitions of the experiment, the occurrence pattern of control messages oscillates. This oscillation is due to the occurrence of control messages inside the system to hold the basic rule of DSM production system. In addition, it can be due to the occurrence of an event inside a local operating system that is out of control of DSM system.

Using regression to calibrate results, $Size\ of\ Control\ Message = 0.1619t^2 + 3.5951t + 54.273$ is reached. In the repeating experiments shown in Figs. 4 and 5, one can find out that regression equations differ in intercept, which is an initial condition. Regarding this fact, the Communication Manager calculates the cost of *Size of Control Message* using its time relation equation. For this to happen, the cost of a single time unit (including energy costs and network costs) for sending a control message when the system is active is calculated, and based on the given equation for control messages cost in DSM production system, decisions can be made.

It should be noted that one of the main challenges of production lines is considering control messages and their role and importance in production systems. However, since they need a short time to fulfill and use the existing executive data transfer platforms in a production system, they are not usually considered as a dependent cost.

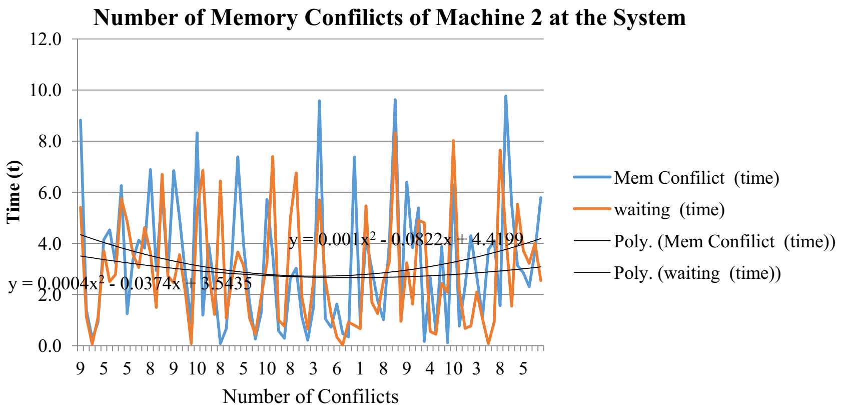

**Fig. 6** The occurrence of data inconsistency in machine 2 and waiting time due to the business of local DSM management and local resource management systems

Figures 4 and 5 indicate that control messages in DSM take longer and a more significant time on message production system. However, this type of messages should be considered in DSM since an effective part of properties of DSM production systems is provided by these messages, i.e., abstraction for users. Thus, they should be regarded as a dependent cost in such production systems.

Another time variable parameter is *Consistency Cost*. In the context of production systems, this parameter has appeared with the name quality of the produced piece; however, in the real world, this parameter is caused by data consistency concept or basic rule of the production line.

In this paper, consistency is assumed as strict consistency. The cost calculation is done via two methods, i.e., whole and single element method of the scheme. When calculating this cost completely, the system method events that violate the basic rule of the production line are collected, and based on the cost of the element, which recovers the system to a consistent state, consistency maintenance cost of the production line is calculated.

In the single element method, for each constituting element of DSM production system, some occurrences that disconnect local element with other elements are counted, and cost of removing these oscillations is calculated. It should be noted that the first method is suitable for computing and processing systems using central machines and the second one is suitable for distributed computing and processing systems.

Matrix multiplication program is executed in case local machines are running activities with high frequency in the modifying memory. The output of the experiment represents the highest limit for the cost of data consistency maintenance in the production line.

Figure 6 shows some occurrences of consistency violations in machine 2 in 100-time units. The dependent variable in this figure shows time spent by memory to recover data and system process inconsistency.

As shown in Fig. 6, upon the occurrence of violation in machine 2, the local memory remains busy for some certain time. During this time, the local memory management performs local operating system's algorithm to recover from inconsistency.

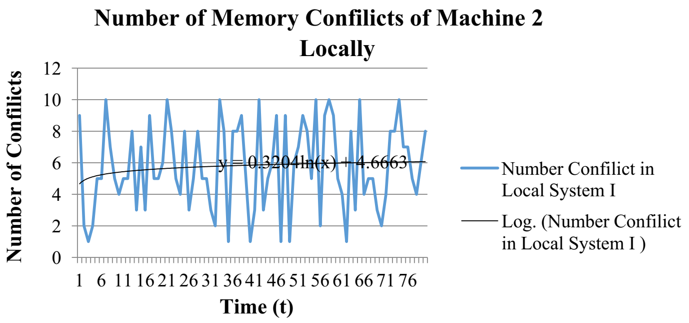

**Fig. 7** Some occurrence of data inconsistency in machine 2

It should be noted that in this case, the reason behind such inconsistency remains unknown to local memory management process. Some process merely reports the occurrence of this inconsistency in machine 2.

The reporting process is a part of DSM management. On the other hand, the inconsistency that occurred in the local machine may lead to a system process, which is a part of the operating system's kernel and is in charge of local machine's resources into the busy state. However, upon activation of either system management units or DSM management unit, the communication manager can calculate conflict cost inside the local machine.

Figure 7 shows the logarithmic relation between occurrences of data inconsistency in machine 2 and system time.

As shown in Fig. 7, there is a logarithmic relation between numbers of occurrences of inconsistency in machine 2 at the time of the system process xxx. Another reason for these oscillations would be called the role of machine 2 in DSM production system and whether this machine has had an activity finishing role in the system or not.

To remove these oscillations, regression is used and finally $conflict = 0.3204ln(t) + 4.6663$ would be produced by the equation, which states the relation between some data inconsistencies and time in machine 2. This equation cannot be considered as part of DSM costs since it has a numerical nature.

On the other hand, in Fig. 7, variable conflict has been shown without considering its role in the cost of using DSM. Communication manager unit can use each a cost factor of DSM management in local machine or factor cost of local machine management factor. In this paper, the cost of DSM management factor has been used on the local machine.

If we take a look at the two existing regression equations in Fig. 6, it is obvious that the difference between two regression equations is equal to $y = 0.8764 - 0.0448x + 0.0006x\hat{}2$. Time factor in mentioned function is cost because the cost factor can be calculated per each time unit, Therefore:

$$0.0006Cost_{Confilict,2} \wedge 2 + 0.0448Cost_{Confilict,2} = 0.3204ln\,(t) + 4.6663 \quad (17)$$

Simplifying Eq. 17, the cost of data consistency maintenance in machine 2 and similarly for any other member machines of the system can be produced.

The other two time variables are non-grasp variables. Using graph theory and by calculating the cost of a path in a graph, the non-grasp variable cost is calculated.

Therefore, each edge in transfer graph is assigned a number called waiting for a cost that shows the cost of idle time of a machine between the two machines. On the other hand, another number called uptime cost is assigned to each edge. This number represents activation cost of DSM between the two machines.

The two mentioned costs of the formation of non-grasp transfer graph are used as the generator cost of this graph. The result of calculation of two mentioned cost is always considered as a fixed number obtained from timing nature of the non-grasp transfer. Although waiting cost and uptime cost are timing nature costs and have different values during the time, they always become a fixed cost after calculation. The reason of such thing in graph and calculation of graph cost in non-grasp transfer and the existence of sigma in non-grasp transfer is due to the change in the mentioned costs to timing fixed costs. Non-grasp transfer cost has been obtained for each edge based on whole costs of waiting time and uptime. If taking a look at computing element nature of the non-grasp transfer, it is clear that these computing elements are kind of fixed costs elements, which are calculated per specified period. Calculating the graph edges cost per specified period showed existence of non-grasp transfer graph that causes cost creation. About non-grasp dependency, the calculation of existed sigma in non-grasp dependency causes the cost of sigma per specified period to change to a fixed cost. An important aspect of two non-grasps is its importance in DSM cost. So, the communication manager unit calculates the possibility of occurrence of non-grasp in DSM, which shows the coefficient of each non-grasp costs in DSM. It is important that communication manager unit calculates DSM costs based on time unit. In other words, this will calculate time cost, about each constituent sub-cost and DSM manufacturing cost. Hence, the cost of each unit is calculated by time, and then functions, which expresses the constituent sub-costs of DSM system cost that is calculated based on time. The reason of this comes from defining the content of variable costs in communication manager unit. So, communication manager unit presents cost function equation as follows:

$$\mathit{Cost\ Function} = \mathit{Fix\ Cost} + \mathit{Cost}(t) \mid t\ \mathit{defines\ for\ each\ sub\ cost\ function} \tag{18}$$

Based on Eq. 18, the communication manager makes decisions about the cost of DSM production system or any other communicating system. On the other hand, upon conversion of the time-independent variable to a time unit and by calculating implicit derivation, communication manager can make decisions about points causing minimum or maximum cost in DSM or costs by which elements increase the total cost of DSM production system.

## 5 Conclusions

This paper presents some mathematical methods to calculate DSM real cost in software DSM on computing environments. The parameters affecting the cost were specified, and a cost model for DSM was proposed.

With this model, DSM real cost and program execution cost in HPC systems can be calculated. This method has obtained a base on the concept of cost in accounting and management sciences (product line cost calculating) to get a real method to calculate the DSM cost.

To make these parameters flexible, parametric weight coefficients were defined, and further, the relations between these parameters were described. Then, the cost/performance ratio can be calculated and used as an estimator to choose suitable DSM mechanisms if costs of other IPC mechanisms are known.

One of the significant challenges in designing and programming of the scientific applications is to calculate program execution cost. Depending on the volume of communication between the processes, in scientific applications, the cost of the underlying communication mechanism is an influential factor in calculating the total cost.

In biological computing systems like Grid and P2P, computing elements are on internet network. System manager does not have any precise vision about computation elements status and how they have IPC. In a time of resource discovery and finding computation element that can meet the requests of global computation activity, if out machine IPC mechanism is expensive, the ratio of cost/performance of such systems become meaningless.

The IPC cost in HPC systems, especially in biological systems like Grid and P2P, is unclear. In computing systems, IPC cost opposites to hardware costs and scientific program creation, and producing costs is a complex that part of it follows the operation cost pattern and heft of it follow the activity cost pattern.

DSM mechanisms are commonly used for transmitting large and complex data structures in scientific and engineering problems.

The nature of the activity and DSM communication cost cause this cost to follow up how the program executes and the performance of environment and system. So, if there would be any vision about the nature of costs of IPC mechanism, system designers could decide about the feasibility of a special kind of HPC system (Cluster, Grid, P2P, and Cloud computing).

In this paper, based on a systematic method and equivalency between production line system and DSM, activity costs and operation costs of DSM are described. According to the selected core activity, activity costs are proposed as false sharing cost, page transfer overhead cost, and paging overhead cost.

These parameters can be considered as DSM abstract features for this purpose, and a method to analyze the effect on DSM real cost is presented by modeling them and finding the costs effects between them, and how that affects the final cost.

Layer pattern is used in the calculation of activity cost related to DSM. This means each of the events causes the basic and core activity getting out of natural and normal mode that is considered as an atomic activity, which, from the perspective of DSM manager, is the only cost of DSM production line manager saving. The activities of the part of DSM manager for bringing back the DSM production line to its natural process are considered for the calculation of exact cost. If the nature of implementation of DSM manager were such that it could manage the events like consistency and false sharing more than two layers, this model would be acceptable.

Based on the nature of activity costs in the extraction of effective factors on cost, the effects of environment and system were taken into consideration. Therefore, in surveying the cost of necessary activities for DSM production line to come back into basic and core activity, the cost of the out of DSM system activities like the cost for process maintenance or the cost of data compatibility is considered suspended.

For increasing the accuracy, the gained costs and the coefficient for the impact of actual situation (condition of running time of program on DSM) are proposed for calculation of those in real time, using time cost (considering a cost for each time unit) and system call (considering cost for executing each system call in operating system level) being used. The information is related to the calculation of DSM cost based on information of data structures of the operating system before extraction and calculation.

Based on the nature differences of closed computing systems like Cluster and biological computing systems like P2P and Grid, the differences of calculation of cost for each of them was proposed when describing the equations. However, because of proposing a general equation, calculation patterns for Cluster systems costs and calculation of the parameters in computing systems have been described. The model is examined by a case study on a distributed P2P system platform.

A special feature of a P2P computing system is the creation of a system based on supply and demand; this feature causes the calculation of real cost (the cost of execution) as DSM is considered as a place for supply and demand. This method of cost calculation causes a pattern for calculating the main coefficient to be presented. This P2P computing system can be used as a cluster computing system because of supporting a concept called area calculation. The process for calculating the cost of using from DSM in a distributed P2P computing system is expressed for examining the model.

The expressed model in this paper provides this ability for management element of a computing system to calculate the cost of using DSM and considering the environmental factors and also inner-system factors on DSM cost parameters that whether they can decide about using DSM for IPCs or not.

## References

1. Expósito, RR et al (2013) Running scientific codes on Amazon EC2: a performance analysis of five high-end instances. J Comput Sci Technol 13:153–159
2. Al Geist, Reed DA (2017) A survey of high-performance computing scaling challenges. Int J High Perform Comput Appl 31(1):104–113
3. Thackston R, Fortenberry R (2015) High performance computing: considerations when deciding to rent or buy
4. Zhang, Z, Cherkasova L, Loo BT (2014) Optimizing cost and performance trade-offs for MapReduce job processing in the cloud. In: 2014 IEEE Network Operations and Management Symposium (NOMS). IEEE
5. Kurmann C, Rauch F, Stricker TM, (2003) Cost/performance tradeoffs in network interconnects for clusters of commodity PCs. Workshop on Communication Architecture for Clusters, Nice, France
6. Rauber T, Rünger G (2013) Parallel programming: for multicore and cluster systems. Springer, Berlin
7. Tootaghaj DZ et al (2015) Evaluating the combined impact of node architecture and cloud workload characteristics on network traffic and performance/cost. In: 2015 IEEE International Symposium on Workload Characterization (IISWC). IEEE
8. Adams M (2014) HPGMG 1.0: a benchmark for ranking high performance computing systems

9. Sukharev PV et al (2017) Benchmarking of high performance computing clusters with heterogeneous CPU/GPU architecture. In: IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). IEEE
10. Dongarra J, Heroux MA, Luszczek P (2015) HPCG benchmark: a new metric for ranking high performance computing systems. Knoxville
11. Al-Roomi M et al (2013) Cloud computing pricing models: a survey. Int J Grid Distrib Comput 6(5):93–106
12. Bhowmick A, Prasad CGVN (2017) Time and cost optimization by grid computing over existing traditional IT systems in business environment. Int J 5:93–98
13. Han R et al (2014) Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Gener Comput Syst 32:82–98
14. Núñez A, Merayo MG (2014) A formal framework to analyze cost and performance in map-reduce based applications. J Comput Sci 5(2):106–118
15. Iosup A et al (2011) Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distrib Syst 22(6):931–945
16. Menascé D, Almeida V (1990) Cost-performance analysis of heterogeneity in supercomputer architectures. In: Proceedings of Supercomputing'90. IEEE
17. Marathe A et al (2013) A comparative study of high-performance computing on the cloud. In: Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing. ACM
18. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29(4):1012–1023
19. De Alfonso C (2013) An economic and energy-aware analysis of the viability of outsourcing cluster computing to a cloud. Future Gener Comput Syst 29(3):704–712
20. Kaplan R, Anderson SR (2013) Time-driven activity-based costing: a simpler and more powerful path to higher profits. Harvard business press, Boston
21. Tahir M et al (2016) Framework for Better Reusability in Component Based Software Engineering. J Appl Environ Biol Sci (JAEBS) 6:77–81
22. Fenton N, Bieman J (2014) Software metrics: a rigorous and practical approach. CRC Press, Boca Raton
23. Berriman GB et al (2010) The application of cloud computing to astronomy: a study of cost and performance. In: Sixth IEEE International Conference on e-Science Workshops. IEEE
24. Deelman E et al (2015) Pegasus, a workflow management system for science automation. Future Gener Comput Syst 46:17–35
25. Yan Z et al (2011) Cloud versus in-house cluster: evaluating Amazon cluster compute instances for running MPI applications. In: State of the Practice Reports. ACM
26. Woitaszek M, Tufo HM (2010) Developing a cloud computing charging model for high-performance computing resources. In: IEEE 10th International Conference on Computer and Information Technology (CIT). IEEE
27. Aviram A et al (2012) Efficient system-enforced deterministic parallelism. Commun ACM 55(5):111–119
28. Otley D, Emmanuel KMC (2013) Readings in accounting for management control. Springer, Berlin
29. Schöner G (2013) Dynamical systems thinking. In: Handbook of developmental systems theory and methodology, p 188
30. Drury CM (2013) Management and cost accounting. Springer, Berlin
31. Deegan C (2012) Australian financial accounting. McGraw-Hill Education Australia
32. Lian X et al (2015) Cache coherence protocols in shared-memory multiprocessors
33. Lenoski DE, Weber W-D (2014) Scalable shared-memory multiprocessing. Elsevier, Amsterdam
34. Qura-Tul FASN, Khan AKDMS (2015) Development of cluster computing—a review. Development 5(1):1–9
35. Satish N et al (2012) Can traditional programming bridge the ninja performance gap for parallel computing applications? ACM SIGARCH Computer Architecture News, vol 40, no 3. IEEE Computer Society
36. Menezo LG, Puente V, Gregorio J-A (2015) Flask coherence: a morphable hybrid coherence protocol to balance energy, performance, and scalability. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE

37. Serrano Gómez M (2013) Scheduling local and remote memory in cluster computers. Dissertation, Editorial Universitat Politècnica de València
38. Behrends R et al (2016) HPC-GAP: engineering a 21st-century high-performance computer algebra system. Concurr Comput Pract Exp 28(13):3606–3636
39. Kasahara H et al (2012) Method for controlling heterogeneous multiprocessor and multigrain parallelizing compiler. US Patent 8,250,548, 21 Aug
40. Marongiu A, Benini L (2012) An OpenMP compiler for efficient use of distributed scratchpad memory in MPSoCs. IEEE Trans Comput 61(2):222–236
41. Engle C et al (2012) Shark: fast data analysis sing coarse-grained distributed memory. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM
42. Cruz EHM et al (2014) Dynamic thread mapping of shared memory applications by exploiting cache coherence protocols. J Parallel Distrib Comput 74(3):2215–2228
43. Habel R, Silber-Chaussumier F, Irigoin F (2013) Generating Efficient Parallel Programs for Distributed Memory Systems. Technical Report CRI/A-523, MINES ParisTech and Télécom SudParis
44. Sim J et al (2012) A performance analysis framework for identifying potential benefits in GPGPU applications. ACM SIGPLAN Notices, vol 47, no 8. ACM
45. Kaashoek MF (2015) Parallel computing and the OS. SOSP History Day 2015. ACM
46. Bericht T, Darmstadt TH, Informatik F, Theel OE, Fleisch Br D (1996) A dynamic coherence protocol for distributed shared memory enforcing high data availability at low costs. IEEE Trans Parallel Distrib Syst 7(9):915–30
47. Yuan D et al (2014) Simple testing can prevent most critical failures: an analysis of production failures in distributed data-intensive systems. In: 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)
48. Medya S, Cherkasova L, Magalhaes G, Ozonat K, Padmanabha C, Sarma J, Sheikh I (2016) Towards performance and scalability analysis of distributed memory programs on large-scale clusters. In: Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, pp 113–116
49. He S et al (2013) A cost-aware region-level data placement scheme for hybrid parallel i/o systems. In: IEEE International Conference on Cluster Computing (CLUSTER). IEEE
50. Susmit B (2014) The software architecture for efficient distributed interprocess communication in mobile distributed systems. J Grid Comput 12(4):615–635
51. Sharifi M, Mirtaheri SL, Khaneghah EM (2010) A dynamic framework for integrated management of all types of resources in P2P systems. J Supercomput 52(2):149–170
52. Khaneghah EM (2017) PMamut: runtime flexible resource management framework in scalable distributed system based on nature of request, demand and supply and federalism. US Patent 9,613,312, 4 Apr
53. Musial P, Nicolaou N, Shvartsman AA (2014) Implementing distributed shared memory for dynamic networks. Commun ACM 57(6):88–98
54. Kim J, Vaidya NH (1997) A cost model for distributed shared memory using competitive update. In: Fourth International Conference on High-Performance Computing, Bangalore, India
55. Gray J (1988) The cost of messages. In: Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing, Toronto, Ontario, Canada
56. Kim J-H, Vaidya NH (1997) A cost model for distributed shared memory using competitive update. In: Proceedings of Fourth International Conference on High-Performance Computing. IEEE
57. Li S et al (2015) An extensible framework for predictive analytics on cost and performance in the cloud. In: International Conference on Cloud Computing and Big Data (CCBD). IEEE
58. Dave VS, Dutta K (2014) Neural network based models for software effort estimation: a review. Artif Intell Rev 42(2):295–307
59. Hassan HA, Mohamed SA, Sheta WM (2016) Scalability and communication performance of HPC on Azure Cloud. Egypt Inform J 17(2):175–182
60. Midgley G (ed) (2003) Systems thinking. Sage, London
61. Thüm T et al (2014) A classification and survey of analysis strategies for software product lines. ACM Comput Surv (CSUR) 47(1):6
62. Metzger A, Pohl K (2014) Software product line engineering and variability management: achievements and challenges. In: Proceedings of the on Future of Software Engineering. ACM

63. Sharifi M, Tirado-Ramos A, Khaneghah EM, Mirtaheri SL (2010) Formulating the real cost of dsm-inherent dependent parameters in HPC clusters. In: SMTP workshop in conjunction with the IEEE International Parallel & Distributed Processing Symposium (IPDPS 2010), 19 April
64. Power R (2014) Abstractions for in-memory distributed computation. Dissertation, New York University
65. Vasava, HD, Rathod JM (2015) Software based distributed shared memory (DSM) model using shared variables between multiprocessors. In: International Conference on Communications and Signal Processing (ICCSP). IEEE
66. Maosen H, Wei H, Huang Y (2016) Enabling mobile device coordination over distributed shared memory. In: IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). IEEE
67. Pelley S, Chen PM, Wenisch TF (2014) Memory persistency. In: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). IEEE
68. Alglave J, Maranget L, Tautschnig M (2014) Herding cats: modelling, simulation, testing, and data mining for weak memory. ACM Trans Program Lang Syst (TOPLAS) 36(2):7
69. Ghosh S (2014) Distributed systems: an algorithmic approach. CRC Press, Boca Raton
70. Kaxiras S et al (2015) Turning centralized coherence and distributed critical-section execution on their head: a new approach for scalable distributed shared memory. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing. ACM
71. Das D, Ray RS, Ray UK (2016) Implementation and consistency issues in distributed shared memory. Int J Comput Sci Eng 4(12):125
72. Dulloor S R et al (2014) System software for persistent memory. In: Proceedings of the Ninth European Conference on Computer Systems. ACM
73. Low Y et al (2014) Graphlab: a new framework for parallel machine learning. arXiv:1408.2041
74. Javanbakht Z, Öchsner A (2017) Introduction to Marc/Mentat. In: Advanced finite element simulation with MSC Marc. Springer, Cham
75. Shrivastava A et al (2016) Automatic management of software programmable memories in many-core architectures. IET Comput Digit Tech 10(6):288–298