Journal of Communications Technology, Electronics and Computer Science, Issue 4, 2016
ISSN 2457-905X

Query Based Resource Allocation and Performance Prediction in Distributed Database

S. Jagannatha, T. V. Suresh Kumar, K. Rajanikanth
Department of Computer Applications
M S Ramaiah Institute of Technology
Bangalore, India City, Country
jagannatha@msrit.edu, tvsureshkumar@msrit.edu, rajanikanth@msrit.edu

***Abstract*—Resource allocation is one of the main issues for providing services in an efficient way on distributed databases. Improving the performance is one of the key research issues by proper design of efficient distributed database and proper use of resources in information technology. The cost of each computational service depends on the amount of computations. The system resources have to be allocated in order to handle the workload and minimize the cost of computing by using proper allocation strategy. Performance is strongly related to the allocation of resources and data fragments in a distributed environment. The query requires data to be accessed from one or multiple sites. The required data are fragmented and placed in various data centers, where resources exist. In this paper, we propose an algorithm to minimize the total data-transfer cost required for processing the queries by proper allocation of resources. We are finding the optimum allocation strategy, which predicts the performance by estimating the cost.**

Keywords—Distributed database, Resource sharing, UML;

I. INTRODUCTION

Developing distributed database systems by fragment design and resource allocation based on query requirement is a more challenging issue. Distributed processing is an effective way to improve the performance of a database system. Performance is a very important issues in distributed database system. Performance issues, addressing are often discussed separately from Software Development Life Cycle (SDLC). Localization of data fragments, avoids remote data accessed by reducing the data exchange between sites; resource usage is a key issue to be addressed.

Database design involves making decisions on the placement of fragments across the sites of a computer network based on query and correspondents fragments required. Data fragments are stored in different data centers that are connected over a communication network. If the fragments and resources are allocated properly, then the problems can be solved by parallel computing. Query based resource allocation strategy for query processing that maximizes the use of resources and minimizes the cost of query computation. The proposed methodology helps to improve the performance in distributed databases by query based resource allocation [1], [2].

In order to predict the performance of the database in early design stages, we must address the different issues: Distributed query, Distributed update propagation, Distributed concurrency control, load balancing, replication, Distributed catalog management. In a company, the database organization might reflect the organizational structure, which is distributed into units. Data fragment can be achieved by developing a distributed database system which makes data accessible by all units' stores data close to where it is most frequently used.

Resource allocation is one of the main issues in solving database applications. The goal is to minimize the total data-transfer cost required for processing the queries by proper allocation of resources. The optimum resource allocation strategy for query processing is determined. Data fragments are distributed geographically over the set of resources. Performance is strongly related to the query assignment, allocation of resources and data fragments in a distributed environment. The query requires data to be accessed from one or many sites. The use case is executed by various combinations of servers.

Data fragments are allocated based on the query assigned into the resources to avoid remote data access. Hence system resources have to be allocated to handle workload and minimize the cost of computing and maximize the utility of resources. The proposed algorithm determines the optimal allocation strategy that minimizes the cost of computation. Finally, allocation spaces can be searched using standard algorithms like Backtracking, Branch-and-Bound, etc. to obtain the optimal allocation.

II. RELATED WORK

In [3] Jonathan M. Graham describes the efficient techniques which are needed to allocate resources across distributed sites, in order to minimize processing costs and to improve response times for users. In [4] author establishes formal measurements for provisioning of virtualizes resources under and over in infrastructures, specifically for Software-as-a-Service (Seas) platform deployments. The author proposes a resource allocation model to deploy SaaS applications over computing platforms by taking into account their multi tenancy for creating a cost-effective, scalable environment and also the author proposes policies that allocate the resources if available, otherwise the request is rejected, and the request is placed in a FIFO queue.

In [5], we address the concepts of resource allocation in cloud computing services. It is proposed how effectively we can utilize the resource allocation by computing parallel applications. The cost of each resource is computed based on its utilization. The approach uses game-theoretic approaches for optimization of resources versus cost. Constructing non dominated local coteries to solve the problem in a distributed way reduces communication overheads. The algorithm achieves the highest degree of fault-tolerance in resource allocation problem.

Dynamic resource allocation and reallocation of geographically distributed resources are the main challenges inherent to the resource allocation process, offering a stepwise modeling phase to the optimization phase proposed in [6]. In [1] author introduces the resource allocation algorithm for computing the distributed computing system into account of heterogeneity of the computational resources. The algorithm resolves single point failures and manages remote tasks and local tasks. Allocations based on resource's requirements and communication cost for overall task response time is considered. Daniel A. Menasc in [7] describes servers of the data centers dynamically deployed among the various application environments using analytic queuing network models with different workload.

Suchita Upadhyaya proposed task allocation algorithm in DDB, based on proper input fragmentation, distribution, transformation of cost matrices, communication network details, topology, location of the nodes, allocation of computer resources, and selection of link capabilities. The inputs are also included as data fragmentation type, an amount of data replication based on the node requirements, and operating strategies: query execution and concurrency control issues, which produce optimum allocation nodes and the results are recorded in cost computation matrix in[8].

In [9] Adnene Guabtni proposes an approach that combines proprietary cloud based load balancing techniques and density-based partitioning query processing across relational database as a service in cloud computing environments conducted over a real-world data. The proposed approach is implemented and tested as a multi-layer web application and database layer. In [10] the author proposes a policy that allocates the resources if availthe requesttherwise the rauest is rejected, and request is placed in an FIFO queue. It uses the resource leases as resource allocation abstraction and implements these leases by allocating Virtual Machines. Proposed dynamic planning based scheduling algorithm is implemented in that which can admit new leases and prepare the schedule whenever a new lease can be accommodated, and it maximizes resource utilization and acceptance of leases.

In [11] resource allocation mechanism based on reverse auction and proposes a market model based on reverse auction and presents the bidding process of reverse auction. It also provides optimal resource provider for each consumer. In automatic resource allocation strategy, based on a market mechanism a provider is proposed. Market models of ARAS-M are constructed, in requirements of the Client. A Genetic Algorithm based automatic price adjusting algorithm is introduced. Experiment results show that it approximately achieves the equilibrium state and are capable of achieving resource balance in resource usage. In [12] we study the resource allocation at the applthe authorn for better resource utilization, and author proposed a threshold-based dynamic resource allocation scheme that dynamically allocates the virtual resources among the load changes which then can use the threshold method to optimize the decision of resource reallocation. Allocation and reallocation techniques for resources are used in a distributed database system, which partitions data between a set of nodes in a network. It proposes multiple token borrowing policies which anticipate future need and keep the system balanced.

In [13] the author describes a distributed solution which integrates workload prediction and distributed non-linear optimization techniques. The On-demand access from computing resources is also used, which enable application providers to scale their services. This work attempts to establish formal measurements for under, and over provisioning of virtualizes resources described in [5]. In [14], two proactive resource allocation algorithms for fault-tolerant asynchronous real-time distributed systems are discussed. In [15] the author proposed an algorithm of job scheduling based on Berger's model. The algorithm establishes the dual fairness constraint. The general expectation functions in accordance with the classification of tasks. The second constraint is a resource fairness, justice function that is used to judge the fairness of the resource allocation.

In [16] Javier Espadas attempts to establish a formal measurement for provisioning of virtualizes resources under and over infrastructures, specifically for Software as Service platform deployments. The author proposes a resource allocation model to deploy applications over computing platforms by taking into account their multi tenancy for creating a cost-effective scalable environment. In [17] author considers a resource allocation problem which is local in the sense that the maximum number of users competing for a particular resource at any time instant is bounded and also at any time instant the maximum number of resources that a user is willing to get is bounded.

III. System Models

Most of the applications in a distributed database system are resource sharing in nature. The requests received in the system are processed by a set of resources. The resources parallel executes the set of queries based on the allocation of task or sub task in a distributed system [1]. In order to model this process, in the Table I, system is considered as set of computing resources. Let R_1, R_2… R_n is the set of computational resources that share N number of tasks S_i. Each task Si consists of sub tasks, which can process parallel. Task Si assigning available resources Rj. When multiple subtasks Si are assigned to resources Rj, they proportionally share the Rj's capacity and expense.

TABLE I. RESOURCE SHARING MATRIX

	R1,	R2,	R3	…	Rn
S1	0	1	1	…	1
S2	0	0	1	…	1
S3	1	0	0	…	0
			……		
S_n	1	0	1	…	1

Finally, Table I represents the resource sharing model of DDS Applications. Rows in the table represent the tasks to be performed in DDS applications and columns in the table represent the computational resources in the Distributed system architecture. Elements (i, j) represent 1 or 0. 1 indicates the subtasks in the task Si assign to the resources Rj; zero indicates task not assigned to any resource.

IV. PROPOSED METHODOLOGY

A. Methodology

The performance of a distributed database is strongly related to the fragment allocation in the nodes of a computer network. A query requires data to be accessed from one or more sites. The cost of computing the query depends on the fragment allocation and the query allocation of the resources. The goal is to minimize the total data transfer and resource usage cost for transaction processing. The cost can be minimized by proper allocation of resources and data availability in distributed environment [13], [37], [38].

B. Prolbem Definition

We have proposed a methodology to assess the performance in distributed databases by allocating the fragments of the databases for various resources based on a given query [1]-[6], [37]-[38]. Let us assume that there are n use cases and m servers. The data are assumed to be fragmented and replicated over them servers. Depending upon the actual fragmentation and replication, we will have certain choices in allocating servers to the use cases. Each such allocation incurs certain cost and would result in certain completion time for the use case. Our objective is to allocate servers to the use cases so as to reduce the costs and completion time considering all use cases of the applications. Allocation of servers to realize a given use case is an important issue in reducing communications cost and improving efficiency.

C. Algorithm for Resource Allocation

Algorithm to determine the optimal allocation by resource allocation is as follows:

1) Develop a use case model of a given DDS application.
2) Let us assume that there are n use cases and m servers in a given architecture. The data is assumed to be fragmented and replicated over the m servers.
3) Let us assume that server k, k = 1, 2 … m; has an associated cost of $C_{k,}$ It is shared by all the use cases, which make use of that server proportionately.
4) Each use case U_k may have V_k server combinations that can execute it. We assume that we can estimate the time for completing U_k with each of these server combinations; the costs of Uk can be obtained by summing up the cost of the servers in the combinations V_k. Where U_k is the kth use case and V_k is the combination of servers that execute U_k
5) Compute the Completion Time Matrix(CTM).
6) The cost of a Server is also shared by the use case which is executing on the particular server.
7) Compute the turnaround time for all the use cases.
8) From the Cost Matrix derived above, compute the total cost of use case Uk as the sum of costs in row k of the matrix.
9) Assume that the turnaround time is weighted by W_t and Cost by Wc. Compute the effective cost for U_k by
10) W_t * (turnaround time of U_k) + (W_c) * (total cost for U_k).
11) Compute the average cost of the allocation.

D. Problem Scenario

Each Ui use case can have Qi queries that may be computed in parallel. These queries are sharing Rj resources and share the Rj's capacity and expense Pj according to its processing capacity. Each sub query may be waiting to share Rj. The assigning of these queries into the resources based on the data fragments available. A solution of the scheduling problem is a non-negative matrix using allocation matrix and to an execution time matrix to compute the total cost incurred in each use case and its resource utilization. The allocation matrix is used to represent the assignment of the query and allocation of a fragment to the resources that are required by the queries of each use case. The execution matrix gives the time required to process the corresponding query of the use case in the respective resource. The utility of the scenarios of the use case Ui is calculated. The objective of each use case is to maximize its utility.

The entry a_{ij} specifies the sub task of the use case Ui assigns to the resources Rj. Using allocation matrix another two matrices are obtained: Completion time matrix T, and final execution matrix E. Let t_{ij} of T be the turnaround time it takes a resource R_j to complete a_{ij} queries of the use case $U_i,\ldots,U_i$ queries are parallel and dependent, the completion time of use cases U_i is given by max {tij | tij Εti}, where ti denotes the vector of the ith row of matrix T. The entry Eij of the matrix E is the network cost for using a resource Rj to complete aij

queries. So the total cost of usage of Ui is $\sum_{j=1}^{m} e_{ij}$. Completion time and cost of usage of resources of each use case U_i is a tradeoff. Let wt, and we be the weight of completion time and network cost respectively. Therefore the

Total cost of

$$U_i = w_t \cdot \max_{t_{ij}} \in \{t_{ij}\} + w_e \cdot \sum j e_{ij} \quad \ldots \qquad (1)$$

V. Resource Allocation Illustration

In this chapter, we propose a new approach for server allocation as described below, along with an illustrative example of a case study:

Let us assume that there are n use cases and m servers. The data is assumed to be fragmented and replicated over the m servers. Depending upon the actual fragmentation and replication we will have certain choices in allocating servers to the use cases. Each such allocation incurs certain cost and would result in certain completion time for the use case. Our objective is to allocate servers to the use cases so as to reduce the costs and completion times considering all use cases of the applications.

Case Study

We consider a case study of an Airline Reservation Application. The customers make a reservation online using a browser. Nachtfliegen airline system has major functions: Flight Booking, Login, Abandon, Get Flight planning, Find Flights, Select Seat, Get Fare, Purchase Itinerary, and Store Itinerary, the database is fragmented and deployed in a given architecture of eight resources. The details in[1][6] presents the typical requests from the users for the application may be authenticating the user, getting the page of the application, searching for the appropriate flights, selecting the desired seat, purchase the ticket, store the details about the flight for later reference, and abandon the Itinerary. The details can be found in[1], [6].

Number of use cases, n = 8

1) Let the number of server's m = 8

Let us assume that server k, k = 1, 2 … m; has an associated cost of Ck which is shared by all the use cases which make use of that server proportionately.

Case Study (continued):

Cost vector = [5, 6, 3, 4, 9, 8, 5, 4]

2) Each use case Uk may have Vk server combinations that can execute it. We assume that we can estimate the time for completing Uk with each of these server combinations; the costs of U_k can be obtained by summing up the cost of the servers in the combinations Vk. Where Uk is the kth use case, and Vk is the combination of servers that execute Uk

Case Study (continued):

Let V1 = 2; Thus use case U1 can be executed by 2 combinations of various servers available, i.e. combination 1 is $\{S_1, S_2, S_3, S_4\}$ and combination 2 is $\{S_2, S_5, S_8\}$ Let the completion times with the second combination be {20, 18, 10}. Under the assumption that the servers are executed U1 only. We assume similar data for all use cases, and the corresponding server combinations are available. Thus, the total number of possible server allocations is V1 x V2 x V3 x … x Vn With a specific choice, we get specific Server Allocation Matrix (SAM). Example is shown below:

TABLE II. Server Allocation Matrix(SAM)

	S1	S2	S3	S4	S5	S6	S7	S8
U1	0	1	0	0	1	0	0	1
U2	1	0	1	0	1	0	1	0
U3	1	1	0	0	1	1	0	0
U4	1	1	1	1	0	1	0	0
U5	1	0	1	1	1	0	1	0
U6	1	1	1	0	0	1	1	1
U7	1	1	1	0	1	0	0	0
U8	0	0	1	1	1	1	1	0

Elements (i, j) represent 1 or 0 in table II, 1 indicates the subtasks in the task U_i assign to the resources R_j; zero indicates task not assigned to any resource. For the above SAM, use cases U_1 under the assumption that the Servers are executing U_1 only are as follows: U_1: Allocation is $\{S_2, S_5, S_8\}$; $U_{2:}$ Allocation is {S1, S3, S5, S7}, Similarly other use cases.

3) Now, we need to compute the Completion Time Matrix(CTM) as follows:

The completion times in Step 3 above assumes that the servers were executing a single-use case. However, the Servers may be executing more than one use case concurrently as indicated in the SAM above. For example, S8 is executing both U1 and U6. Thus, the execution times will proportionately increase.

Case Study (continued):

For the above SAM, the execution times for the use cases U_1 under the assumption that the Servers are executing U_1 only are as follows:

U_1: Allocation is $\{S_2, S_5, S_8\}$; Execution Times: [S_2: 4; S_5: 3; S_8:5} Using similar data corresponding to other use cases, we get the following matrix of execution times under the assumption of no concurrency:

TABLE III. EXECUTION TIME MATRIX WITH NO SERVER SHARING

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
U_1	-	4	-	-	3	-	-	5
U_2	4	-	5	-	4	-	6	-
U_3	6	4	-	-	8	7	-	-
U_4	3	6	4	2	-	8	-	-
U_5	4	-	6	7	2	-	4	-
U_6	3	4	5	-	-	6	2	8
U_7	6	7	3	-	8	-	-	-
U_8	-	-	5	6	7	3	4	-

The new Execution Time Matrix (ETM) with the Servers executing multiple use cases will be follows:

TABLE IV. EXECUTION TIME MATRIX OF WITH SERVER SHARING

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
U_1	-	20	-	-	18	-	-	10
U_2	24	-	30	-	24	-	24	-
U_3	36	20	-	-	48	28	-	-
U_4	18	30	24	6	-	32	-	-
U_5	24	-	36	21	12	-	16	-
U_6	18	20	30	-	-	24	8	16
U_7	36	35	18	-	48	-	-	-
U_8	-	-	30	18	42	12	16	-

4) The cost of a Server also is shared by the use cases that are executing on that particular server.
Case Study (continued):
Thus, the cost matrix corresponding to the above SAM will be as follows:

TABLE V. COST MATRIX FOR THE SAM

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
U_1	-	24	-	-	27	-	-	20
U_2	20	-	15	-	36	-	30	-
U_3	30	24	-	-	72	56	-	-
U_4	15	36	12	8	-	64	-	-
U_5	20	-	18	28	18	-	20	-
U_6	15	24	15	-	-	48	10	32
U_7	30	42	9	-	72	-	-	-
U_8	-	-	15	24	63	24	20	-

5) From ETM derived in Step 4, we obtained turnaround time for all the use cases as follows:
Turnaround time of U_k is the maximum value in row k of ETM.
Case Study (continued):
The vector of turnaround times = [27, 36, 72, 64, 28, 48, 72, 63]

6) From the Cost Matrix derived above, we compute the total cost of use case U_k as the sum of costs in row k of the matrix.
Case Study (continued):
The total cost of use case U_k = [71, 101, 182, 135, 116, 144, 153, 144]

7) The usefulness of a particular allocation for a specific use case depends both on the turnaround time for that use case and the total cost for that use case. Assume that the turnaround time is weighted by W_t and Cost by W_c. Then the effective cost for U_k will be
W_t x turnaround time of U_k + W_c x total cost for U_k.
Thus, for a given allocation, we can compute the effective cost for each use case.
Case Study (Continued):
Assuming that $W_t = W_c = 0.5$,
Effective cost for the use cases corresponding to the above SAM shown in Step 3 above is: [46, 66, 114, 84, 70, 87, 101, 94].
The average cost of the allocation = average of the costs for the use cases= 82.875
Effective cost for the use cases corresponding to the Server Allocation prescribed in Table II presented in the Fig. 1

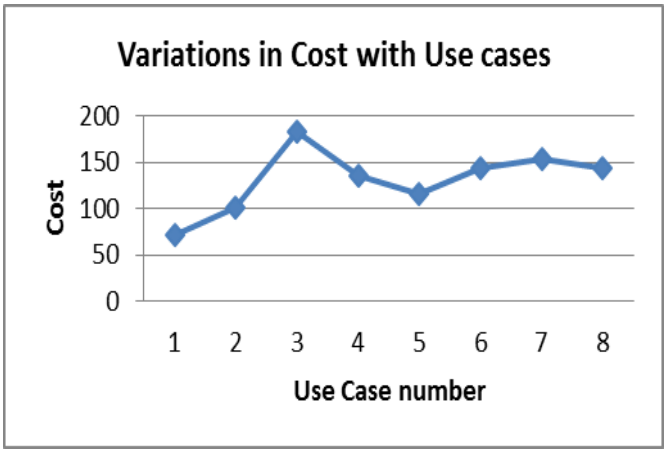

Fig 1 Variations in Cost with Use Case

The graph in Fig. 2 below shows the variations in turnaround time with use cases for one such Server allocation in Table I.

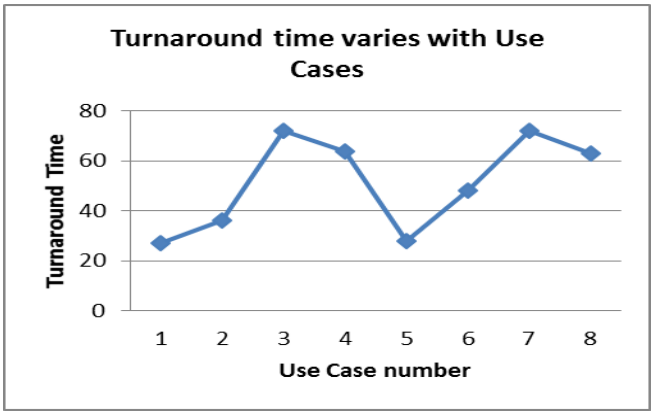

Fig 2 Turnaround Time with Use Case

Fig. 3 below show's variations in effective cost with use cases i.e., when one subtask assigns to one resource.

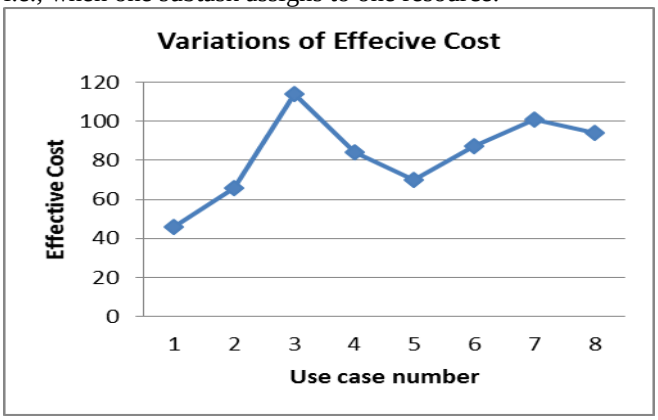

Fig 3 Variations in Effective Cost with Use Case

From the Fig. 1, 2 and 3 presents, the results of one particular server allocation (Table II) in computing DDS application. However, the results are compared with different allocation of servers, and the results are extended with sensitivity analysis. Now, the allocated spaces can be searched using standard algorithms like Backtracking, Branch-and-Bound, etc. to obtain the optimal allocation.

VI. Optimal Search

Let V_1 = 2; Thus use case U_1 can be executed by 2 combinations of various servers available, i.e. combination 1 is {S_2, S_3, S_6, S_8} and combination 2 is {S1, S_2, S_4, S_5, S_8}.Similarly, U2 has V2 = 1, U3 has V3=1, U4 has V4= 2, U5 have V5=1, U6 has V6 =1, U7 has V7 = 2 and U8 has V8 = 2 server combinations. Thus, the total number of possible server allocations is V_1 x V_2 x V_3 x … x V_n i.e. 16 server combinations. We obtained turnaround time for all the use cases presented in Table VI.

TABLE VI. Turnaround times for 16 ways of allocation of servers

Allocation No	**Turnaround time for all the use cases**							
1.	27	36	72	64	28	48	72	63
2.	35	25	55	74	52	45	45	28
3.	44	21	26	75	46	58	52	28
4.	29	45	46	48	48	45	42	32
5.	35	44	52	42	38	62	75	85
6.	28	45	72	75	77	36	25	42
7.	44	54	54	54	82	73	65	49
8.	72	54	25	65	35	53	32	45
9.	56	54	58	74	75	42	46	45
10.	45	45	56	47	85	59	65	63
11.	42	48	75	50	56	54	58	57
12.	56	53	54	52	54	54	54	54
13.	54	50	41	53	36	29	27	45
14.	82	29	35	34	38	62	75	42
15.	46	45	24	58	83	54	43	56
16.	72	54	52	45	46	46	35	58

The turnaround time obtained in Table VI indicates that there are sixteen different ways are possible for resource allocation. The example in the last row of Table VI the turnaround time is obtained in one such resource allocation [72, 54, 52, 45, 46, 46, 35, and 58]. We compute the total cost of use case Uk as the sum of costs in row k of the matrix of all 16 different ways of server allocation. The total cost of each use case presented in the Table VII.

TABLE VII. TOTAL COST FOR ALLOCATION OF SERVERS IN DIFFERENT WAYS

Allocation Number	**Total cost of use case**							
1.	133	125	123	145	135	125	156	175
2.	189	98	98	145	112	102	105	108
3.	105	123	145	185	175	165	140	153
4.	140	126	142	150	146	165	164	125
5.	185	129	158	147	186	190	170	160
6.	145	135	154	152	142	134	120	150
7.	95	90	98	99	125	145	420	430
8.	152	152	175	156	153	145	145	189
9.	156	148	196	136	154	123	147	185
10.	125	145	198	145	163	158	148	148
11.	185	165	145	189	178	185	182	163
12.	158	145	146	153	185	187	123	154
13.	156	184	156	132	145	157	163	182
14.	145	154	152	178	121	141	143	153
15.	148	173	192	151	162	151	173	125
16.	142	152	143	152	169	145	142	135

The total cost of all the sixteen possible allocation assignments to the resource is presented in the Table VII. Example in the last row of Table VII7 the cost is obtained in one such resource allocation [142, 152, 143, 152, 169, 145, 142, 135].

Particular allocation for the specific use case depends both on the turnaround time for that use case and the total cost for that use case. Then the effective cost for Uk will be

W_t x turnaround time of U_k + W_c x total cost for U_k.
Assuming that $W_t = W_c = 0.5$.
The average cost of the allocation = average of the costs for the use cases presented in the Table VIII. Least average cost of effective cost of implementing use case is 51.13. Hence optimum allocation of servers presented in the Table IX.

TABLE VIII. THE EFFECTIVE COST FOR THE USE CASES CORRESPONDING TO THE SAM

Allocation Number	**Effective cost for U_k**								**Average**
1.	78	45	52	65	112	53	54	52	63.88
2.	53	62	54	53	56	54	58	55	55.63
3.	42	44	45	68	112	44	96	114	70.63
4.	121	65	85	95	83	75	86	83	86.63
5.	63	52	54	55	45	63	45	45	52.75
6.	45	52	84	75	96	63	52	45	64.00
7.	55	55	62	63	45	45	52	82	57.38
8.	45	53	66	52	45	55	48	85	56.13
9.	56	53	54	56	42	58	87	75	60.13
10.	52	54	54	52	45	62	45	45	51.13
11.	65	63	58	57	78	82	47	92	67.75
12.	48	75	52	75	76	75	92	154	80.88
13.	75	52	54	56	58	53	57	55	57.50
14.	85	75	72	48	82	75	77	82	74.50
15.	45	45	54	41	46	85	77	52	55.63
16.	54	58	85	54	55	56	55	41	57.25

The optimal allocation obtained through complete search is: average is cost 51.13.

TABLE IX. OPTICAL ALLOCATION

	S_1	**S_2**	**S_3**	**S_4**	**S_5**	**S_6**	**S_7**	**S_8**
U_1	0	1	0	0	1	0	0	1
U_2	1	0	0	0	1	0	1	0
U_3	1	1	0	0	0	1	0	0
U_4	0	1	0	1	0	1	0	0
U_5	1	0	1	0	0	0	1	0
U_6	0	1	1	0	1	0	0	1
U_7	0	0	1	0	0	0	1	0
U_8	0	0	0	1	0	0	1	1

The total number of possible server allocations is V1 x V2 x V3 x … x Vn . i.e. 16 server combinations. We computed the total cost, turnaround time and effective cost of all 16 possible servers' combinations. From the results, we get the optimal allocation obtained through complete search is presented in the Table IX.

VII. Conclusion and Future Work

In this paper, we use resource allocation strategies in minimizing the cost of computation in distributed database applications during the preliminary design stages. We propose an algorithm for optimal allocation strategy with respect to fragments and resources that maximum utilization of resources. The fragment allocation and resource allocation strategy are considered in minimizing the cost computation. One Server allocation strategy is illustrated by the case study and presented the result.

Various combinations of server's allocation are also presented. Each combination of servers is illustrated by determining the effective cost of computation, total cost of the allocation of servers, and turnaround time for all the combinations of servers. Based on the results obtained, choose the optimum allocation of server combinations is presented. Results are obtained for various governing parameters.

References

[1] Reza Basseda "Fragment Allocation in Distributed Database Systems Faculty of Electrical and Computer Eng., School of Engineering, University of Tehran Database Research Group 2006

[2] Ajit M. Tamhankar and Sudha Ram "Database Fragmentation and Allocation: An Integrated Methodology and Case Study" IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 28, NO. 3, MAY 1998

[3] Jonathan M. Graham Norfolk State University Norfolk, Virginia U.S.A. "Theoretical properties of two problems of distribution of interrelated data" ACM SE'06 March 10-12, 2006, Melbourne, Florida, USA.

[4] Michele Mazzuccoa,b, Dmytro DyachukcA Optimizing Cloud providers revenues via energy efficient server allocation 2210-5379/$ – see front matter © 2011 Published by Elsevier Inc. doi:10.1016/j.suscom.2011.11.001

[5] X You, J Wan, X Xu, C Jiang, W Zhang and J Zhang. ARAS-M: Automatic Resource Allocation Strategy based on Market Mechanism in Cloud Computing Environment. Journal of Computers, 6(7):1287, July, 2011.

[6] S. Jagannatha, D.E.Geetha, Dr. T. V. Suresh Kumar, Dr Rajanikanth, "Load Balancing in Distributed Database System using Resource Allocation Approach", Proc. International Journal of Advanced Research in Computer and Communication Engineering, Volume 2, Issue 7, pp. 2529-2535.

[7] Mohamed N. Bennani and Daniel A. Menasc´eResource Allocation for Autonomic Data Centers using Analytic Performance Models 2005 Intl Conf on Autonomic Computing, Seattle Washington June 13-16, 2005.

[8] Dr. Suchita Upadhyaya and Suman Lata Task allocation in Distributed computing VS distributed database systems : A Comparative study IJCSNS International Journal of Computer Science and Network Security, VOL.8 No .3, March 2008.

[9] Adnene Guabtni · Rajiv Ranjan · Fethi A. Rabhi A workload-driven approach to database query processing in the cloud Springer Science+Business Media, LLC 2011

[10] M N Bennani and D A Menasc´e. Resource Allocation for Autonomic Data Centers using Analytic Performance Models. In Proceedings of the International Conference on Autonomic Computing, pages 229-240, 2005.

[11] X Wang, J Sun, H Li, C Wu and M Huang. A Reverse Auction Based Allocation Mechanism in the Cloud Computing Environment. Applied Mathematics Information Science, 7(1):75-84, 2013.

[12] A Nathani, S Chaudharya, G Somanib. Policy Based Resource Allocation in Iaas Cloud, Elsevier, 2011.

[13] A G R Ranjan and A Fethi. Workload-Driven Approach to Database Query Processing in the Cloud, Springer, 2011.

[14] D Ardagna, S Casolari, M Colajanni, B Panicuccia and Dual. Time-Scale Distributed Capacity Allocation and Load Redirect Algorithms for Cloud Systems .Journal on Parallel Distributed Computing, 72:796–808, 2012.

[15] B Ravindran, P Li, and T Hegazy. Proactive Resource Allocation for Asynchronous Real-Time Distributed Systems in the Presence of Processor Failures. Journal of Parallel and Distributed Computing, 63:1219–1242, 2003

[16] Javier Espadas a, Arturo Molina b, Guillermo Jiménez a, Martín Molina b, Raúl Ramírez a, David Conchaa A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved. doi:10.1016/j.future.2011.10.013

[17] S F El-Zoghdy, M Nofal, M A Shohla and A El-sawy. An Efficient Algorithm for Resource Allocation in Parallel and Distributed Computing Systems. International Journal of Advanced Computer Science and Applications, IJACSA, 4(2):251-259, 2013.

[18] S Upadhyaya and S Lata. Task allocation in Distributed Computing vs Distributed Database Systems: A Comparative study. International Journal of Computer Science and Network Security, 8(3): 338-346, March, 2008.

[19] Connie U. Smith1 and Lloyd G. Williams2 SOFTWARE PERFORMANCE ENGINEERING 1 Performance Engineering Services, PO Box 2640, Santa Fe, NM 87504, www.perfeng.com2, Software Engineering Research, 264 Ridgeview Lane, Boulder, CO 80302, (303) 938-9847

[20] Bente Anda, Hege Dreiem, dag I. K Sjobergand Magne Jorgensen, "Estimating Software development Effort based on Use Cases Experiences from Industry", www.idi.ntnu.no/emner/tdt4290/docs/fagliglumI2001 -anda.pdf.

[21] Connie U. Smith, Performance Engineering of Software Systems, Reading, MA, Addison-Wesley,1990.

[22] Ayman Issa (1,2), Mohammed Odeh (1,2), & David Coward (2) Software Cost Estimation Using Use-Case Models: a Critical Evaluation 0-7803-9521-2/06/$20.00 §2006 IEEE

[23] Connie U. Smith1 and Lloyd G. Williams2 SOFTWARE PERFORMANCE ENGINEERING 1Performance Engineering Services, PO Box 2640, Santa Fe, NM 87504, www.perfeng.com 2Software Engineering Research, 264 Ridgeview Lane, Boulder, CO 80302, (303) 938-9847

[24] D.C. Petriu, H Shen, "Applying the UML Performance Profile: Graph Grammar-based derivation of LQN models from UML specifications", in Computer performance Evaluation-Modeling techniques and Tools, (T.Fields, P.Harrison, J.Bradley, U.Harder, Eds.) LNCS 2324, pp.158-177, Springer, 2002.

[25] Dorin Petriu, Murray Woodside: "Analysing Software Requirements Specifications for Performance", Proceedings of the 3rd international workshop on Software and performance 2002, Rome, Italy July 24 - 26, 2002, pp.1-9.

[26] Connie U. Smith and Lloyd G. Williams, Performance Solutions, 2000.

[27] Kahkipuro P, "UML based performance modeling framework for object oriented distributed systems", Proceedings of second international

conference on Unified Modeling Language. October 1999, USA, (Springer Verlag, LNCS) pp. 1723

[28] Shahin Kamali Pedram Ghodsnia Khuzaima Daudjee "Dynamic Data Allocation with Replication in Distributed Systems" 978-1-4673-0012-4/11/$26.00 ©2011 IEEE 189.

[29] Lloyd G. Williams and Connie U. Smith, "PASA: A Method for the Performance

[30] Assessment of Software Architectures", WOSP '02, July 24-26, 2002 Rome, Italy, pp. 179 – 189.

[31] Robert T. Futrell, Donald F. Shafer, and Linda I. Shafer: "Quality Software Project Management", Pearson Eduacation, 2006.

[32] Simonetta Balsamo Roberto Mamprin Moreno Marzolla: "Performance Evaluation of Software Architectures With Queuing Network Models", 2004.

[33] Simonetta Balsamo Moreno Marzolla, "Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models", Proceedings of the 5th international workshop on Software and performance, 2005, Palma, Illes Balears, Spain July 12 - 14, 2005, pp. 37 – 42.

[34] Lloyd G. Williams and Connie U. Smith, "PASA: A Method for the Performance Assessment of Software Architectures", WOSP '02, July 24-26, 2002 Rome, Italy, pp. 179 – 189.

[35] J Espadas, A Molina, G Jiménez, M Molina, R Ramírez and D Conchaa. A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures, Elsevier, 2011.

[36] G Wei, A V Vasilakos, Y Zheng and N Xiong. A game-theoretic method of Fair Resource Allocation for Cloud Computing Services. Springer, 2009.

[37] P T Endo, A V de Almeida Palhares, N N Pereira, G E Goncalves, D Sadok, J Kelner, B Melander and J E Mangs. Network, Resource Allocation for Distributed Cloud: Concepts and Research Challenges, IEEE, 25(4):42 – 46, 2011.

[38] S. Jagannatha, T. V. Suresh Kumar, K. Rajani Kanth, "Algorithm of Performance Prediction by Resource Sharing in Distributed Database Systems," Proc. International Journal of Computer Applications, Volume 66, No.11, 2013, pp. 5-11.