Replicas Redistribution in Distributed Database using Creative Evolutionary Systems Approach

AL-DAHOUD ALI AND MOHAMED A. BELAL
Faculty of Science and Information Technology
Al-Zaytoonah University
Jordan

Abstract

In Distributed Data Bases (DDB), the replication of partitions into multiple hosts could be helpful for reducing the average remote access requests; however, there is a trade-off between replicating partitions into multiple hosts and maintaining remote requests. If we increase the number of replicas, the remote requests will be decreased, on the other hand, redistributing partitions will cost both space and time.

In this paper, a system is proposed that is capable of evolution of a wide range of data redistributing schemes, in Distributed Database DDB, from scratch, using the Creative Evolutionary Systems (CES) approach, by evolution of such data replication schemes, we can search for near optimal design of these schemes. A new formulation of the problem of data redistributing in DDB is given, a novel genotype to phenotype, and crossover operation, are provided. Finally, the performance of the proposed model is compared with different replicas redistribution schemes and the results are discussed and analyzed.

Key-Words: - Creative Evolutionary System, Distributed Data Bases, Distributed Systems.

1. Introduction

Storing data in distributed database systems can be done in one of the following way: replication, fragmentation, and combination of both. In the fragmentation approach, the data is partitioned into several parts; each part is stored at different sites [14].

Distributed database system consists of loosely coupled sites that share no physical components. Furthermore, the database systems that run on each site may have a substantial degree of mutual independence. The data in distributed database can be stored in several ways [14, 15,17,18,19,20, and 21]: Firstly, replica: the system maintains many copies of the data. Each copy is stored at a different site. Secondly, partition: the data is divided into several parts. Each part is stored at a different site. Finally, partition and replication: the data is partitioned into many parts. The system maintains many copies of each part.

Several approaches have been proposed to solve the problem of migrate and/or replicate data in distributed databases, such as central, migration, and the full replication algorithm [22]. In this paper, we will study the use of creative evolutionary systems approach to search for optimal distribution of replicas in the distributed database environment.

Creative Evolutionary System (CES) [5] is used to explore the space of possible solutions, by using the concept of evolution; these solutions could be out of the designer imagination and may be better. CES has been used successfully in evolving novel designs in many applications, and shown impressive results in a large set of areas.

The concept of using evolution for exploration, rather than optimization, was, first, introduced by Bentley [4]; since then, researchers have been interested in using computer systems to aid creativity in design [5]. Such as in nature, evolution creates populations that exist in dynamic and interacting environment where it is possible to explore possibilities and creative solutions [8]. Before that time, Frazer had used evolutionary algorithms to evolve architecture systems [6], applications of CES includes industrial designs, conceptual design, arts, music composition, digital circuits, fighter pilot strategies, graphics [5], and conceptual blending [7], evolutionary arts

[11,12], One of the applications is designing flying objects by folding sheets of paper [3].

This approach is required to automate stages of the design process, in our work, this system is decided to evolve and create the structure of partitions replications in a distributed database from scratch. Such system would demonstrate the feasibility of using creative design to solve the problem of task allocation or data redistribution in distributed database systems.

2. Related work

Many algorithms have been proposed for distributed database, the algorithms can be categorized by whether they migrate and/or replicate data. In central algorithm, a special site contains the entire data. It acts as a data server for other sites. It services the read requests from other sites by returning the required data items to them. On write requests, it updates the data, and returns acknowledgment messages to the site. Duplicate writes requests can be detected by associating sequence numbers with write requests.

In the migration algorithm [16], the data is shipped to the location of the data access request allowing subsequent accesses to the data to be performed locally. The migration algorithm allows only one node to access a shared data at a time. This is a single reader/single writer protocol, since only the threads executing on one host can read or write a given data item at any time [16]. Typically, the whole database or block containing the data item migrates instead of an individual item requested. This algorithm takes advantage of the locality of reference exhibited by programs by amortizing the cost of migration over multiple accesses to the migrated data. However, this approach is susceptible to thrashing, where databases frequently migrate between nodes while servicing only a few requests. To locate a data block, the migration algorithm can make use of a server that keeps track of the location of databases, or through hints maintained at nodes. These hints direct the search for a database toward the node currently holding the database. Alternatively, a query can be broadcasted to locate a database. One disadvantage of the migration algorithm is that only the threads on one host can access data contained in the same block at any given time.

The full replication algorithm [20] allows multiple nodes to have both read and write access to shared data blocks. Because many nodes can write shared data concurrently, the access to shared data must be controlled to maintain its consistency. One possible way to keep the replicated data consistent is to globally sequence the write operations. A simple strategy based on sequencing uses a single global gap-free sequencer which is a process executing on a host participating in DSM. When a process attempts a write to shared memory, the intended modification is sent to the sequencer. This sequencer assigns the next sequence number to the modification with this sequence number to all sites. Each site processes broadcast write operations in sequence number order. When a modification arrives at a site, the sequence number is verified as the next expected one. If a gap in the sequence numbers is detected, either a modification was missed or a modification was received out of order, in which case a retransmission of the modification message is requested. In effect, this strategy implements a negative acknowledgment protocol [16].

3. The Creative Evolutionary System (CES)

In order to enable creativity and design space exploration, constraints must be removed or relaxed [9]; this relaxation can be done interactively [13], anther factor is to let parameters and representation as generic as possible [8], so removal of constraints should be not only in fitness function but also in the problem representation. When the parameters of the representations define a set of components, the ability of the CES to explore the design space increases dramatically [10].

In order to apply CES to a certain application, the phenotype (allowed search space) must be specified, and then the genotype (coding method) should be defined [1]. In Creative Evolutionary Systems, a structured representation of genotype is defined, this data structure is mapped to a phenotype (or a design), and the design space of phenotype should be generic in order to enable the system to propose unexpected results.

The main difference between standard GA and CES is that CES evolves design, or phenotype, from scratch instead of optimizing existing designs; in other words, GA optimizes a set of predefined parameters of a specific design, and CES searches for the optimal design configuration of the problem at hand. In CES, the phenotypes are never manipulated directly; just the coded genotypes are manipulated. The genotypes consist of genes that are arranged in a structure that reflects a generic design solution. Both mutation and crossover operation must be newly defined in order to avoid meaningless solutions.

The CES algorithm is more advanced than the standard GA, one feature of CES is the use of external population to keep best solutions extracted from the internal population; in which the evolution procedure continues to produce candidates for optimal solutions [1], other feature is the use of an explicit mapping from genotype to phenotype. The external population can give a 'life span' time in order to prevent CES from keeping poor solutions that can corrupt the final candidate list.

The mapping form genotype to phenotype in CES is not straightforward, as in standard GA, this process resemble nature, since the DNA is not evaluated directly, rather it follows a set of 'instructions' in order to 'generate' a phenotype [1]; this process enforces the rules of actual object formation and representation. The phenotype of each individual is then evaluated to get its fitness based on the application and environment requirements.

4. The proposed Model

Let us assume that the cost of redistributing a partition is C_1, the cost of a remote request is C_2, and the cost of replicating a request is C_3. We assume the existence of a monitor that is activated to keep track of partitions usage by all hosts, the monitoring is done at discrete times T = 1,2,..,k. Let us assume that m_{ij} is the number of requests of partition j and host i till a time k. The usage of partitions by a matrix $M=[m_{ij}]_{i=1,..,n\,;\,j=1,..,m}$ at time k.

A general partition distribution, at time T = k, can be formulated as a matrix D = $[d_{ij}]$, if d_{ij} = 1 then partition j will be, if not presented, redistributed to host i, otherwise if d_{ij} = 0 then it will not be redistributed.

Each chromosome *v* is represented as a matrix D. However, As a data structure representation, the distribution matrix of each individual v can be modeled as a list l_v of sets s_{vj} {j = 1,2,..,m}, where s_{vj} contains the hosts that hold the partition j as suggested by individual *v*.

The fitness or evaluation function f(D) can be represented as

$$f(D)=C_1\sum_{j=1}^{m}\sum_{i=1}^{n}U(d_{ij}(k)-d_{ij}(k-1))+C_2\sum_{j=1}^{m}\sum_{i=1}^{n}C(d_{ij}(k))m_{ij}+C_3\sum_{i=1}^{n}((\sum_{j=1}^{m}d_j(k)-1)\sum_{j=1}^{m}m_{ij}) \quad (1)$$

Where C(x) function is the complement function that is defined as follows:

$$C(x)=\begin{cases}1 & \text{if } x=0\\ 0 & \text{if } x=1\end{cases}$$

& U(x) function is the unit step function, which is defined as follows:

$$U(x)=\begin{cases}1 & \text{if x} \geq 0\\ 0 & \text{if } x<1\end{cases}$$

The first term in the fitness function represents the cost of replicating the partitions into multiple hosts; on the other hand, the second term stands for the cost of manipulating requests from hosts that do not hold a replication of the partitions of these requests, whereas the third term corresponds to the cost of replicating *write requests* into multiple hosts that contain replicas of the corresponding partitions.

4.1 Population initialization

Each chromosome of an individual v is initialized as follows:

Each s_{vj} is initialized by a random number of hosts; this random number N is not uniformly distributed, rather it follows exponential distribution and it belongs to the range [1,*n*], i.e. it is most probable for the set to have less items. After determining the number of hosts N at each set, the elements of the set will be chosen based on roulette wheel selection, in which hosts with high access requests are more likely to be chosen, and the weight of each selected host will be eliminated from the roulette wheel; in order to not choose this host again.

4.2 Crossover operation in the proposed model

The pair of parents $\{p_1,p_2\}$ is chosen randomly from the population, the new pair of offspring $\{o_1,o_2\}$are formed as follows:
For each set s_{p1j} of the first parent, an host is chosen randomly and assigned to the corresponding set of the offspring s_{o1j}**,** the same operation is repeated for parent s_{p2j} and offspring s_{o2j}. After that, the rest of the sets s_{p1j} and s_{p2j} are combined to form a temporary set s_t. The rest of the set s_{o1j} is formed by scanning s_t, and choosing each host with a probability = 0.5, and the same is done for the set s_{o2j}.

4.3 Mutation operation in the proposed model

The mutation operation is simply done, with a small mutation probability P_m, at each set s_{o1j} or s_{o2j}, by choosing an arbitrary element of it and then altering it to an host that does not exist in the original string.

5. Simulations and results

The proposed model was simulated, and the CES part was tested to verify the significance of our approach. The model was compared with three other approaches namely: The central approach, the fully replication approach, and the greedy approach, the greedy approach was presented in [22]. The parameters of the simulated system were given the following values:

n = 25 nodes
m = 30 partitions
C1 = 500 units
(The unit represents the cost of transferring a block of information through the network)

C2 = 10 units
C3 = 20*0.1 units
(This factor represents the cost of one replication of a write request multiplied by the ratio of write requests in all requests).

The initial partitions distributions over nodes $d_{ij}(k)$ were assumed to be 1 if (j-1) mod n + 1= i, otherwise it equals zero. The usage of partitions or requests (M or [m_{ij}]) are initialized by random integers that are uniformly distributed in the closed range [0,600]. The central strategy uses node number one as a central node.

Figure (1) shows the performance of our CES strategy compared to other strategies, the extreme cases of centralization strategy and fully replication are expected to be worse than any heuristic strategy since they are limiting boundaries of the cost of remote access requests and the cost of replication respectively, the greedy strategy has better performance since it assigns the partitions of maximum usage to the corresponding with a minimal overhead of replication. On the other hand, the proposed model compromises between the overhead of replication and the cost of remote access, it took 50 generations in order to reduce the cost of the greedy strategy to 69.3% of its value. The proposed model suggests the redistribution of additional 14 replicas compared to the greedy strategy that helps reducing the overall expected cost. Therefore, we can conclude that the CES approaches balances between the desire of replicating partitions in order to minimize the remote access requests (as in the fully replication approach), and the conservation of replicating partition in order to minimize the overhead of replication (as in the central approach).

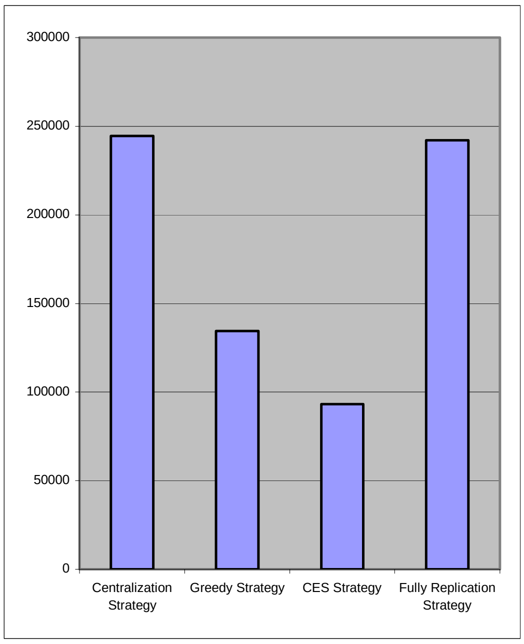

Figure (1): The cost of CES strategy compared to other strategies

Figure (2) illustrates the effect of partition size (C_1/C_2) over the performance of each algorithm, the CES has better performance at the different values of partition size, whereas the centralized algorithm performs badly at high partition size, similarly fully replicated methods is worse at high partition size.

6. Conclusions

In this paper, a new approach for data redistribution in distributed systems has been introduced, this approach is based on creative evolutionary systems CES for searching for the optimal solution or configuration that minimizes the cost of replication versus remote requests. The proposed approach has proved its capability to find a sub-optimal solution at different environments, such as the partition's size and the distribution of partitions requests.

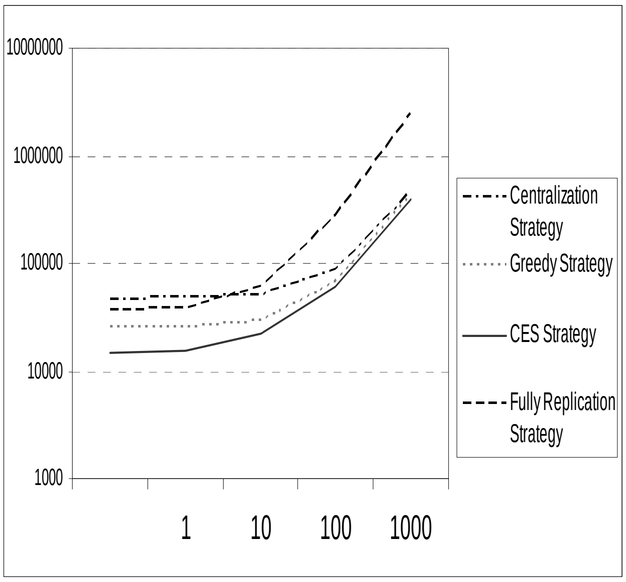

Figure (2): The effect of partition size over each method

References:

[1] Bentley, P. J. & Wakefield, J. P. 1997. Generic Evolutionary Design. Chawdhry, P.K., Roy, R., & Pant, R.K. (eds) *Soft Computing in Engineering Design and Manufacturing*. Springer Verlag, Part 6, 289-298.

[2] Chvál, J.: L-systems Based Generative Mapping in the Evolutionary Design. In: Sinčák, P., Kvasnička, V., Pospíchal, J., Kelemen, J., Návrat, P. (eds.): Slovensko-České rozpravy o umelej inteligencii (Kognícia a umelý život III), Košice, 2003, pp. 309-314. ISBN 80-89066-64-X.

[3] Divina, D. Edwards, S. Kain Creative Evolution of Flying Objects. In Proceedings of CCIA International Conference 2003.

[4] P. J. BENTLEY, *Evolutionary Design by Computers*, Morgan Kaufmann Publishers Inc., ISBN 1-55860-605-X (1999).
[5] P. J. BENTLEY AND D. W. CORNE, *Creative evolutionary systems*, Morgan Kaufmann Publishers Inc.,2001.
[6] Frazer J.H., An Evolutionary Architecture, Architectural Association, London, 1995 (out of print)
http://www.aaschool.ac.uk/publications/ea/intro.html
[7] Theresa Gartland-Jones ,"Visual Blends: A Computational System Exploring, Digital Creative spaces". Proceedings of COSIGN-2004, University of Split (Croatia), September 2004.
[8] P. J. Bentley, T. G. W. Gordon, J. Kim and S. Kumar, "New Trends in Evolutionary Computation", Proceedings of the IEEE Congress on Evolutionary Computation, 2001
[9] Bentley, P. J. (1999a). Is Evolution Creative? In P. J. Bentley and D. Corne (Eds) *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems* (CES). Published by The Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB), pp. 28-34.
[10] Bentley, P. J. (2000). Exploring Component-Based Representations - The Secret of Creativity by Evolution? In ACDM 2000, April 26th - 28th, 2000, University of Plymouth.
[11] Rowbottom, A. (1999). Evolutionary Art and Form. In Bentley, P.J. (Ed.) *Evolutionary Design by Computers*. Morgan Kaufman Publishers Inc., San Francisco, CA.
[12] Celestino Soddu, *Generative Art*, www.celestinosoddu.com, 2001.
[13] Hideyuki Takagi, "Interactive Evolutionary Computation: Fusion of Cababilities of EC Optimization and Human Evaluation".
[14] M.T. Ozsu and P. Valduriez, "Principles of Distributed Database Systems", Prentice-Hall, 2nd

Ed. 1999.
[15] Thomas Seidmann, "Distributed Shared Memory Using The .NET Framework", Slovak University of Technology, seidmann@dcs.elf.stuba.sk

[16] CNE Tutorial Modules Central, "Distributed Shared Memory",http://www.cne.gmu.edu/modules/DSM Implementation Related Issues.htm.
[17] M.T. Ozsu and P. Valduriez, "Principles of Distributed Database Systems", Prentice-Hall, 2nd Ed. 1999.

[18] Shigeru Imafuku_ Kazuhiko Ohno Hiroshi Nakashima, "Reference Filtering for Distributed Simulation of Shared Memory Multiprocessors", *Proc. 34th Annual Simulation Symp.*, pp. 219–226, IEEE Computer Society, April 2001.

[19] Bernd Dreier, Markus Zahn, Theo Ungerer, " The Rthreads Distributed Shared Memory System", University of Augsburg, Institute of Informatics, D-86135 Augsburg, Germany, fdreier,zahng@Informatik.Uni-Augsburg.DE.

[20] Michael Stonebraker, Paul M. Aoki, Avi Pfeffer, Adam Sah,Jeff Sidell, Carl Staelin and Andrew Yu, " MARIPOSA: A Wide-Area Distributed Database System", Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720-1776, mariposa@postgres.Berkeley.EDU.

[21] Michael Stonebraker, Paul M. Aoki, Witold Litwin, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, Andrew Yu, " Mariposa: a wide-area distributed database system", The VLDB Journal (1996) 5: 48–63.

[22] Al-Dahoud Ali, "Data Redistributing in Distributed Computing Systems", ICENCO 2004-1st INTERNATIONAL COMPUTER ENGINEERING CONFERENCE, December 27-30 2004 Cairo, Egypt.