International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:10, 2007

Replicating Data Objects in Large-scale Distributed Computing Systems using Extended Vickrey Auction

Samee Ullah Khan and Ishfaq Ahmad

***Abstract*—** This paper proposes a novel game theoretical technique to address the problem of data object replication in large-scale distributed computing systems. The proposed technique draws inspiration from computational economic theory and employs the extended Vickrey auction. Specifically, players in a non-cooperative environment compete for server-side scarce memory space to replicate data objects so as to minimize the total network object transfer cost, while maintaining object concurrency. Optimization of such a cost in turn leads to load balancing, fault-tolerance and reduced user access time. The method is experimentally evaluated against four well-known techniques from the literature: branch and bound, greedy, bin-packing and genetic algorithms. The experimental results reveal that the proposed approach outperforms the four techniques in both the execution time and solution quality.

***Keywords*—**Auctions, data replication, pricing, static allocation.

I. Introduction

DATA object replication techniques determine how many replicas of each objects are to be created, and to which sites they are to be assigned. Such replica schemas critically affect the performance of the distributed computing system (e.g. the Internet), since reading an object locally is less costly than reading it remotely [1]. Therefore, in a read intensive network an extensive replica schema is required. On the other hand, an update of an object is written to all, and therefore, in a write intensive network a constricted replica schema is required. In essence, replica schemas are strongly dependent upon the read and write patterns for each object [2]. Recently, a few approaches on replicating data objects over the Internet have been proposed in [3]–[6] and [7]. The majority of the work related to data replication on the Internet employs the site based replication. As the Internet grows and the limitations of caching become more obvious, the importance of object based replication, i.e., duplicating highly popular data objects, is likely to increase [8].

In this paper, the replica schemas are established in a static fashion. The aim is to identify a replica schema that effectively minimizes the object transfer cost. We propose a novel technique based on the extended Vickrey auction [9], where the players compete for memory space at sites so that replicas can be placed. This approach is compared against four well-known techniques from the literature: genetic [1], branch and bound [6], bin-packing [6], and greedy [9] algorithms. Experimental results reveal that this simple and intuitive approach outperforms the four techniques in both execution time and solution quality.

The remainder of this paper is organized as follows. Section II provides motivation to study the object replication problem and encapsulates the related work. Section III formulates the object replication problem (ORP). Section IV concentrates on modeling the auction mechanism for the ORP. The experimental results and concluding remarks are provided in Sections V and VI, respectively.

TABLE I
NOTATIONS AND THEIR MEANINGS

Symbol	Meaning
M	Total number of sites in the network.
N	Total number of objects to be replicated.
O_k	k-th object.
o_k	Size of object k.
S_i	i-th site.
s_i	Size of site i.
r_k^i	Number of reads for object k from site i.
R_k^i	Aggregate read cost of r_k^i.
w_k^i	Number of writes for object k from site i.
W_k^i	Aggregate write cost of w_k^i.
NN_k^i	Nearest neighbor of site i holding object k.
$c(i,j)$	Communication cost between sites i and j.
P_k	Primary site of the k-th object.
R_k	Replication schema of object k.
$C_{overall}$	Total overall data transfer cost.
ORP	Object replication problem.
EVA	Extended Vickrey auction.

II. Motivation and Related Work

A. Motivation

Caching attempts to store the most commonly accessed objects as close to the clients as possible, while replication distributes a site's contents across multiple mirror servers. Replication accounts for improved end-to-end response by allowing clients to download from their closest mirror server. Caching can be viewed as a special case of replication when mirror servers store only parts of a site's contents [3]. This

Manuscript received June 1, 2005.

Samee Ullah Khan is with the Department of Computer Science and Engineering, University of Texas at Arlington, TX 76019 USA (phone: 817-272-3607; fax: 817-272-3784; e-mail: sakhan@cse.uta.edu).

Ishfaq Ahmad is with the Department of Computer Science and Engineering, University of Texas at Arlington, TX 76019 USA.

analogy leads to some interesting comparisons. For instance, cache replacement algorithms are examples of on-line, distributed, locally greedy algorithms for data allocation in replicated systems. Furthermore, caches do not have full server capabilities and thus can be viewed as a replicated system that sends requests for specific object types (e.g., dynamic pages) to a single server. Essentially, every major aspect of a caching scheme has its equivalent in replicated systems, but not vice versa. Replication, as a side effect, leads to load balancing and increases client-server proximity [9].

As a rule of thumb, a replica placement technique should pursue the following line of action.

1. Determine the network topology.
2. Specify the objects that are to be replicated.
3. Obtain the access frequencies of the objects. The access frequencies are either known *apriori* or determined using some prediction techniques.
4. Based on the above information, employ an *algorithmic technique* to replicate objects based on some *optimization criteria* and *constraints*.
5. Finally, determine a redirection method that sends client requests to the best *replicator* that can satisfy them.

Based on the above passage, an effective replica placement technique determines the replica allocation which gives the highest data accessibility in the whole network. If the network topology is comprised of M sites which are connected (directly or indirectly) to each other and N denotes the number of data objects that are specified for replication, then, the number of possible combinations of replica allocation is expressed by the following expression:

$$\left\{ N! / \left(N - C \right)! \right\}^{M},$$

where C is the overall memory capacity of M sites.

In order to determine the optimal allocation among all possible combinations, we must analytically find a combination which gives the highest data accessibility considering the following parameters:

1. Access frequencies from each site to each data object.
2. The probability that each site's memory capacity remains unchanged.
3. The probability that the network connectivity remains unchanged.

Even if some looping is possible the computational complexity is very high, and this calculation must be done every time when either of the above three parameters change. Moreover, among the above three parameters, the later two cannot be formulated in practical because they follow no known phenomenon.

For these reasons, we take the following approach:

1. Replicas are relocated in a specific period (*relocation period*).
2. At every relocation period, replica allocation is determined based on the access frequency from each site to each data object and the network topology at the moment.

Based on this approach we propose a game theoretical technique that effectively and efficiently determines a replica schema that is competitive, scalable and simple compared to: GRA [1], Aε-Star [6], and Greedy [9].

B. Related Work

The data replication problem (see Section 3 for a formal description) is an extension of the classical file allocation problem (FAP). Chu [11] studied the file allocation problem with respect to multiple files in a multiprocessor system. Casey [12] extended this work by distinguishing between updates and read file requests. Eswaran [13] proved that Casey's formulation was NP-complete. In [7] Mahmoud et al. provide an iterative approach that achieves good solution quality when solving the FAP for infinite server capacities. A complete although old survey on the FAP can be found in [14].

In the context of the Internet, replication algorithms fall into the following three categories: 1) the problem definition does not cater for the client accesses, 2) the problem definition only accounts for read access and 3) the problem definition considers both read and write access including consistency requirements. These categories are further classified into four categories according to whether a problem definition takes into account single or multiple objects, and whether it considers storage costs.

The main drawback of the problem definition in category 1 is that they place the replicas of every object, in the same node. Clearly, this is not practical, when many objects are placed in the system. However, they are useful as a substitute of the problem definition of category 2, if the objects are accessed uniformly by all the clients in the system and utilization of all nodes in the system is not a requirement. In this case category 1 algorithms can be orders of magnitude faster than the ones for category 2, because the placement is decided once and it applies to all objects.

Most of the research papers tackle the problem definition of category 2. They are applicable to read-only and read-mostly workloads. In particular this category fits well in the context of CDNs. Problem definitions [14]–[18] have all been used in CDNs. The two main differences between them are whether they consider single or multiple objects, and whether they consider storage costs or not. The cost function in [7] also captures the impact of allocating large objects and could possible be used when the object size is highly variable. In [19] the authors tackled a similar problem – the proxy cache placement problem. The performance metric used there was the distance parameter, which consisted of the distance between the client and the cache, plus the distance between the client and the node for all cache misses. It is to be noted that in CDN, the distance is measured between the cache and the closest node that has a copy of the object.

The storage constraint is important since it can be used in order to minimize the amount of changes to the previous replica placements. As far as we know only the works reported in [1] and [20] have evaluated the benefits of taking storage costs into consideration. Although there are research papers which consider storage constraints in their problem definition, yet they never evaluate this constraint (e.g. see [10], [13], [21], and [22]).

Considering the impacts of writes, in addition to that of reads, is important, if content providers and applications are

able to modify documents. This is the main characteristic of category 3. Some research papers in this category also incorporate consistency protocols – in many different ways. For most of them, the cost is the number of writes times the distance between the client and the closest node that has the object, plus the cost of distributing these updates to the other replicas of the object. In [17], [18], [22]–[24], the updates are distributed in the system using a minimum spanning tree. In [10] and [21] one update message is sent from the writer to each copy, while in [1] and [6] a generalized update mechanism is employed. There a broadcast model is proposed in which any user can update a copy. Next, a message is sent to the primary (original) copy holder site which broadcasts it to the rest of the replicas. This approach is shown to have lower complexity than any of the above mentioned techniques. In [23] and [26], it is not specified how updates are propagated. The other main difference among the above definitions is that [1], [5], [6], [22], [23], and [27]–[29] minimize the maximum link congestion, while the rest minimize the average client access latency or other client perceived costs. Minimizing the link congestion would be useful, if bandwidth is scare.

Our work differs from all the above in: 1) describing a problem definition that combines both the server selection and replica placement problems, 2) taking into account the more pragmatic scenario in today's distributed information environments, we tackle the case of allocating replicas so as to minimize the network traffic under storage constraints with "read from the nearest" and "update through the primary server policies, 3) indirectly incorporating the minimization of link congestion via object transfer cost, 4) extensively evaluating the impact of storage constraints similar to the evaluations performed in [1] and [6], and 5) using game theoretical techniques.

Recently, game theory has emerged as a popular tool to tackle optimization problems especially in the field of distributed computing. However, in the context of data replication it has not received much attention. We are aware of only three published articles which directly or indirectly deal with the data replication problem using game theoretical techniques. The first work [27] is mainly on caching and uses an empirical model to derive Nash equilibrium. The second work [20] focuses on mechanism design issues and derives an incentive compatible auction for replicating data on the Web. The third work [31] deals with identifying Nash strategies derived from synthetic utility functions. Our work differs from all the game theoretical techniques in: 1) identifying a non-cooperative priced based replica allocation method to tackle the data replication problem, 2) using game theoretical techniques to study an environment where the agents behave in a selfish manner, 3) performing extensive experimental comparisons with a number of conventional techniques using an experimental setup that is mimicking the Web in its infrastructure and access patterns.

III. Object Replication Problem formulation

Consider a distributed system comprising M sites, with each site having its own processing power, memory (primary storage) and media (secondary storage). Let S_i and s_i be the name and the total storage capacity (in simple data units e.g. blocks), respectively, of site i where $1 \le i \le M$. The M sites of the system are connected by a communication network. A link between two sites S_i and S_j (if it exists) has a positive integer $c(i,j)$ associated with it, giving the communication cost for transferring a data unit between sites S_i and S_j. If the two sites are not directly connected by a communication link then the above cost is given by the sum of the costs of all the links in a chosen path from site S_i to the site S_j. Without the loss of generality we assume that $c(i,j) = c(j,i)$. This is a common assumption (e.g. see [5]–[7], [32]). Let there be N objects, each identifiable by a unique name O_k and size in simple data unites o_k where $1 \le k \le N$. Let r_k^i and w_k^i be the total number of reads and writes, respectively, initiated from S_i for O_k during a certain time period t. This time period t determines when to initiate a replica placement algorithm (in our case the auction mechanism), *i.e.*, relocation period. Note that this time period t is the only parameter that requires human intervention. However, in this paper we use analytical data that enables us to effectively predict the time interval t (see Section V.A for details).

Our replication policy assumes the existence of one primary copy for each object in the network. Let P_k, be the site which holds the primary copy of O_k, *i.e.*, the only copy in the network that cannot be de-allocated, hence referred to as primary site of the k-th object. Each primary site P_k, contains information about the whole replication scheme R_k of O_k. This can be done by maintaining a list of the sites where the k-th object is replicated at, called from now on the *replicators* of O_k. Moreover, every site S_i stores a two-field record for each object. The first field is its primary site P_k and the second the nearest neighborhood site NN_k^i of site S_i which holds a replica of object k. In other words, NN_k^i is the site for which the reads from S_i for O_k, if served there, would incur the minimum possible communication cost. It is possible that $NN_k^i = S_i$, if S_i is a *replicator* or the primary site of O_k. Another possibility is that $NN_k^i = P_k$, if the primary site is the closest one holding a replica of O_k. When a site S_i reads an object, it does so by addressing the request to the corresponding NN_k^i. For the updates we assume that every site can update every object. Updates of an object O_k are performed by sending the updated version to its primary site P_k, which afterwards broadcasts it to every site in its replication scheme R_k.

For the ORP under consideration, we are interested in minimizing the total Replication Cost (RC) (or the total network transfer cost) due to object movement, since the communication cost of control messages has minor impact to the overall performance of the system. There are two components affecting RC. The first component of RC is due to the read requests. Let R_k^i denote the total RC, due to S_is' reading requests for object O_k, addressed to the nearest site NN_k^i. This cost is given by the following equation:

$$R_k^i = r_k^i o_k c\left(i, NN_k^i\right), \tag{1}$$

where $NN_k^i = \{Site\ j \mid j \in R_k \wedge \min\ c(i,j)\}$. The second component of RC is the cost arising due to the writes. Let W_k^i be the total RC, due to S_is' writing requests for object O_k, addressed to the primary site P_k. This cost is given by the

following equation:

$$W_k^i = w_k^i o_k \left(c\left(i, P_k\right) + \sum_{\forall j \in R_k, j \neq i} c\left(NN_k^i, j\right) \right). \quad (2)$$

Here, we made the indirect assumption that in order to perform a write we need to ship the whole updated version of the object. This of course is not always the case, as we can move only the updated parts of it (modeling such policies can also be done using our framework). The cumulative RC, denoted as $C_{overall}$, due to reads and writes is given by:

$$C_{overall} = \sum_{i=1}^{M} \sum_{k=1}^{N} \left(R_k^i + W_k^i \right). \quad (3)$$

Let $X_{ik} = 1$ if S_i holds a replica of object O_k, and 0 otherwise. X_{ik}s define an $M \times N$ replication matrix, named X, with boolean elements. Equation 3 is now refined to:

$$X = \sum_{i=1}^{M} \sum_{k=1}^{N} \begin{bmatrix} \left(1 - X_{ik}\right) \left[r_k^i o_k \min\left\{ c\left(i, j\right) \mid X_{jk} = 1 \right\} \right. \\ \left. + w_k^i o_k c\left(i, P_k\right) \right] + X_{ik} \left(\sum_{x=1}^{M} w_k^x \right) o_k c\left(i, P_k\right) \end{bmatrix}. \quad (4)$$

Sites which are not the *replicators* of object O_k create RC equal to the communication cost of their reads from the nearest *replicator*, plus that of sending their writes to the primary site of O_k . Sites belonging to the replication scheme of O_k, are associated with the cost of sending/receiving all the updated versions of it. Using the above formulation, the ORP can be defined as:

"*Find the assignment of* $0,1$ *values in the X matrix that minimizes* $C_{overall}$, *subject to the storage capacity constraint:* $\sum_{k=1}^{N} X_{ik} o_k \leq s_i \ \ \forall (1 \leq i \leq M)$, *and subject to the primary copies policy:* $X_{P_k k} = 1 \ \ \forall (1 \leq k \leq N)$."

The minimization of $C_{overall}$ has the following two impacts on the distributed system under consideration. First, it ensures that the object replication is done in such a way that it minimizes the maximum distance between the replicas and their respective primary objects. Second, it ensures that the maximum distance between an object k and the user(s) accessing that object is also minimized. Thus, the solution aims for reducing the overall RC of the system. In the generalized case, the ORP is NP-complete [1].

IV. Extended Vickrey Auction (EVA)

A. Setup

In the auction setup each primary copy of an object k is a player. A player k can perform the necessary computations on its strategy set by using the site (where it resides) P_k's processor. At each given instance a (sub)-auction takes place at a particular site i chosen in a round robin fashion from the set of M sites. These auctions are performed continuously throughout the system's life, making it a self evolving and self repairing system. However, for simulation purposes ("cold" network [6]) we discrete the continuum solely for the reason to observe the solution quality.

B. Competitiveness

Extended Vickrey Auction

Initialize:
01 LS, L^i.
02 WHILE $LS \neq$ NULL **DO**
03 **SELECT** $S^i \in$LS /*Round-robin fashion */
04 **FOR** each $k \in O$ **DO**
05 B_k = compute (B_k^i); /*compute the benefit*/
06 **Report** B_k to S_i which stores in array B;
07 **END FOR**
08 **WHILE** $b_i \geq 0$
09 B_k = argmax$_k$(B); /*Choose the best offer*/
10 Extract the info from B_k such as O_k and o_k;
11 $b_i = b_i$-o_k; /*Calculate available space and termination condition*/
12 Payment = B_k; /* Maintain Vickrey payment */
13 **IF** $b_i < 0$ **THEN EXIT WHILE ELSE**
14 $L^i = L^i$ - O_k; /*Update the list*/
15 Update NN^i_{OMAX} /*Update the nearest neighbor list*/
16 **IF** L^i = NULL **THEN SEND** info to M to update $LS = LS$ - S^i;
17 **Replicate** O_k;
18 **END WHILE**
19 S_i asks all successful bidders to pay B_k
20 END WHILE

Fig. 1. Pseudo-code for Extended Vickrey Auction (EVA).

Each player k competes through bidding for memory at a site i. Many would argue that memory constraints are no longer important due to the reduced costs of memory chips. However, replicated objects (just as cached objects) reside in the memory (primary storage) and not in the media (secondary storage) [8], [33]. Thus, there will always be a need to give priority to objects that have higher access (read and write) demands. Moreover, memory space regardless of being primary or secondary is limited.

C. Strategy

Each player k's strategy is to place a replica at a site i, so that it maximizes its (the object's) benefit function. The benefit function gives more weight to the objects that incur reduced RC in the system:

$$B_k^i = R_k^i - \left(\sum_{x=1}^{M} w_k^x o_k c\left(i, P_k\right) - W_k^i \right). \quad (5)$$

The above value represents the expected benefit (in RC terms), if O_k is replicated at S_i. This benefit is computed using the difference between the read and update cost. Negative values of B_k^i mean that replicating O_k, is inefficient from the "local view" of S_i (although it might reduce the global RC due to bringing the object closer to other servers).

The pseudo-code for EVA is given in Fig. 1.

D. The Algorithm

We maintain a list L^i at each server. The list contains all the objects that can be replicated at S_i (*i.e.*, the remaining storage capacity b^i is sufficient and the benefit value is positive). We also maintain a list LS containing all servers that can replicate an object. In other words, $S_i \in$LS if and only if $L^i \neq$ NULL. EVA performs in steps. In each step a server S_i is chosen from LS in a round-robin fashion. Each player $k \in O$ calculates the benefit function of object. The set O represents the collection of players that are legible for participation. A player k is legible if and only if the benefit function value obtained for

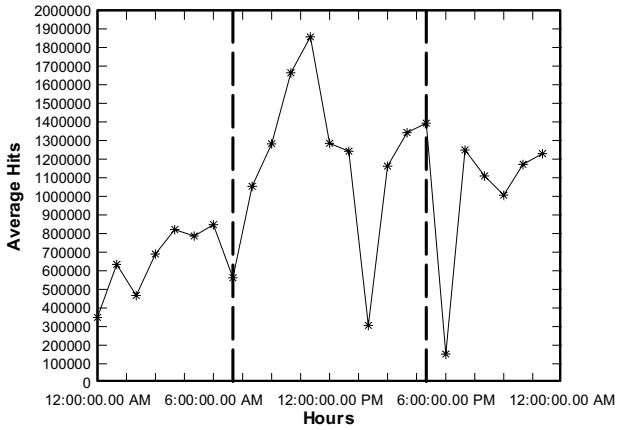

Fig. 2(A). Access on Days when there were no Scheduled Games.

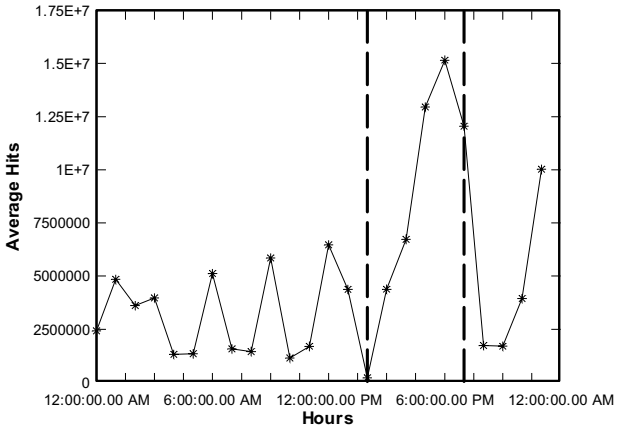

Figure 2(B). Access on Days when there were Scheduled Games.

TABLE II
OVERVIEW OF TOPOLOGIES.

Topology	Mathematical Representation
SGRG [6] (12 topologies)	Randomized layout with node degree (d*) and Euclidian distance (d) between nodes as parameters.
GT-ITM PR [35] (5 topologies)	Randomized layout with edges added between the randomly located vertices with a probability (p).
GT-ITM W [35] (9 topologies)	$P(u,v)=\alpha e^{-d/(\beta L)}$
SGFCGUD [6] (5 topologies)	Fully connected graph with uniform link distances.
SGFCGRD [6] (5 topologies)	Fully connected graph with random link distances.
SGRGLND [6] (9 topologies)	Random layout with link distance having a lognormal distribution [36].

site S_i is the maximum of among all the other benefit function values for sites other than i, *i.e.*, $S_i \geq S_{-i}$. This is done in order to suppress mediocre bids, which, in turn improves computational complexity. It is to be noted that in each step L^i together with the corresponding nearest server value NN_k^i, are updated accordingly.

E. Theoretical Results

Theorem 1: EVA takes $O(MN^2)$ time.

Proof: The worst case execution time of the algorithm is when each server has sufficient capacity to store all objects and the update ratios are low enough so that no object incurs negative benefit value. In that case, the while-loop (02) performs M iterations. The time complexity for each iteration is governed by the for-loop in (04) and the while loop in (08) ($O(N^2)$ in total). Hence, we conclude that the worst case running time of the algorithm is $O(MN^2)$. ■

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

We performed experiments on a 440MHz Ultra 10 machine with 512MB memory. The experimental evaluations were targeted to benchmark the placement policies. The solution quality in all cases, was measured according to the RC percentage that was saved under the replication scheme found by the algorithms, compared to the initial one, *i.e.*, when only primary copies exist.

TABLE III
RUNNING TIME IN SECONDS

Problem Size	Greedy	GRA	Aε-Star	EVA	GMM
M= 500, N= 1350	81.69	117.60	110.46	**78.48**	90.09
M= 500, N= 1400	98.28	127.89	127.89	**81.87**	95.34
M= 500, N= 1450	122.43	139.02	139.02	**87.81**	98.91
M= 500, N= 1500	134.61	148.47	155.40	**90.75**	104.37
M= 500, N= 1550	146.58	168.84	169.47	**95.06**	105.63
M= 500, N= 2000	152.25	177.66	189.21	**105.46**	108.57

A. Relocation Period

As discussed in Sections II.A and III, the time (interval t) when to initiate the EVA requires high-level human intervention. In this section, we will show that this parameter if not totally can at least partially be automated. The decision when to initiate EVA depends on the past trends of the user access patterns. The experiments performed to test the EVA used real user access patterns collected at the 1998 Soccer World Cup website [31]. This access log file has become a "de fecto" standard over the number of years to benchmark various replica placement techniques. Works reported in [6], [9], and [10] all have used this access log for analysis.

Figs. 2(A) and 2(B) show the user access patterns. The two figures represent different traffic patterns, *i.e.*, Figure II(A) shows the traffic recorded on the days when there was no scheduled match, while Fig. 2(B) shows the traffic on the days when there were scheduled matches. We can clearly see that the website incurred soaring and stumpy traffic at various intervals during a 24-hour time period (it is to be noted that the access logs have a time stamp of GMT+1). For example, the days when there was no scheduled match, the traffic was mediocre before 0900 hrs. The traffic increased after 0900 hrs till 2200 hrs. The two vertical dashed lines indicate this phenomenon. These traffic patterns were recorded over a period of 86 days (April 30th 1998 to July 26th 1998). Therefore, on the days when there was no scheduled match, a replica placement algorithm (in our case the EVA) could be initiated twice daily: 1) at 0900 hrs and 2) at 2200 hrs. The time interval t for 0900 hrs would be t = (2200-0900) = 11 hours and for 2200 hrs would be t = (0900-2200) = 13 hours.

On the other hand the days when there were scheduled matches, EVA could be initiated at 1900 hrs and 0100 hrs. It is to be noted that the autonomous agents can easily obtain all the other required parameters (for the ORP) via the user access logs and the underlying network architecture.

B. Experimental Setup

To establish diversity in our experimental setups, the network connectively was changed considerably. In this paper, we only present the results that were obtained using a maximum of 500 sites. We used existing topology generator toolkits and also self generated networks. Table II summarizes the various techniques used to gather forty-five various topologies. All the results reported, represent the average performance over all the topologies.

To evaluate our proposed technique on realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 website [31]. Each experimental setup was evaluated thirteen times, *i.e.*, Friday (24 hours) logs from May 1, 1998 to July 24, 1998. Thus, each experimental setup in fact represents an average of the 585 (13×45) data set points. To process the logs, we wrote a script that returned: only those objects which were present in all the logs (2000 in our case), the total number of requests from a particular client for an object, the average and the variance of the object size. From this log we choose the top five hundred clients (maximum experimental setup), which were randomly mapped to one of the nodes of the topologies. Note that this mapping is not 1-1, rather 1-*M*. This gave us enough skewed workload to mimic real world scenarios. It is also worthwhile to mention that the total amount of requests entertained for each problem instance was in the range of 1-2 million. The primary replicas' original site was mimicked by choosing random locations. The capacities of the sites *C*% were generated randomly with range from *Total Primary Object Sizes/2* to *1.5×Total Primary Object Sizes*. The variance in the object size collected from the access logs helped to instill enough diversity to benchmark object updates. The updates were randomly pushed onto different sites, and the total system update load was measured in terms of the percentage update requests *U*% compared that to the initial network with no updates.

C. Comparative Algorithms

For comparison, we selected four various types of replica placement techniques. To provide a fair comparison, the assumptions and system parameters were kept the same in all the approaches. We chose: 1) from [6] the efficient branch-and-bound based technique (Aε-Star), 2) from [1] the genetic algorithm based technique (GRA) which showed excellent adaptability against skewed workload, 3) from [6] the bin-packing based technique GMM 4) and from [10] the famous greedy approach (Greedy). Due to space limitations, we briefly describe the comparative approaches. Details for a specific technique can be obtained from the referenced papers.

1) Aε-Star

In [6] the authors proposed a 1+ε admissible A-Star based technique called Aε-Star. This technique uses two lists: OPEN and FOCAL. The FOCAL list is the sub-list of OPEN, and only contains those nodes that do not deviate from the lowest cost node by a factor greater than 1+ε. The technique works similar to A-Star, with the exception that the node selection is done not from the OPEN but from the FOCAL list. It is easy to see that this approach will never run into the problem of memory overflow, moreover, the FOCAL list always ensures that only the candidate solutions within a bound of 1+ε of the A-Star are expanded.

2) GMM

In [6] the authors proposed a bin-packing based technique, which we describe as follows: Let O_k and S^i represent the set of objects and sites in the system. Let U be the set of unassigned objects and k be the global minimum of all the replication costs associated with an object. The minimum of such cost as a set $T=min_{0\leq j\leq N-1}(k(O_k,S^i)),\forall O_k \in U$. If during the assignment, the minimum replication cost of an object is the same for two different sites, the tie is broken by the minimum object size. For a node n let *mink(n)* define the minimum element of set T. Thus *mink(n)* represents the best minimum replication cost that would occur if object O_k is replicated to a site S^i, *i.e.*, Global Min-Min (GMM).

3) GRA

In [1] the authors proposed a genetic algorithm based heuristic called GRA. GRA provides good solution quality, but suffers from slow termination time. This algorithm was selected since it realistically addressed the fine-grained data replication using the same problem formulation as undertaken in this article.

4) Greedy

We modify the greedy approach reported in [10] to fit our problem formulation. The greedy algorithm works in an iterative fashion. In the first iteration, all the M sites are investigated to find the replica location(s) of the first among a total of N objects. Consider that we choose an object j for replication. The algorithm recursively makes calculations based on the assumption that all the users in the system request for object j. Thus, we have to pick a site that yields the lowest cost of replication for the object j. In the second iteration, the location for the second site is considered. Based on the choice of object j, the algorithm now would identify the second site for replication, which, in conjunction with the site already picked, yields the lowest replication cost. Observe here that this assignment may or may not be for the same object j. The algorithm iterates forward till either one of the ORP constraints are violated.

D. Results and Discussions

Table III (best times shown in bold) shows the algorithm execution times. The number of sites was kept constant at 500, and the number of objects was varied from 1350 to 2000. With maximum load (2000 objects and 500 sites), the proposed technique EVA saved approximately 50 seconds of termination time then the third fastest algorithm (Greedy).

Superiority of execution time comes at the cost of loss in solution quality. However, EVA showed high solution quality. First, we observe the effects of system capacity increase. An increase in the storage capacity means that a large number of objects can be replicated. Replicating an object that is already extensively replicated, is unlikely to result in significant traffic

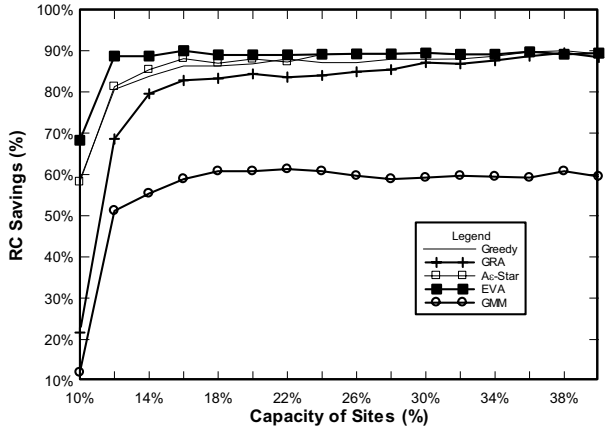

Fig. 3. RC Savings Versus System Capacity (N=2000, M=500, U=5%).

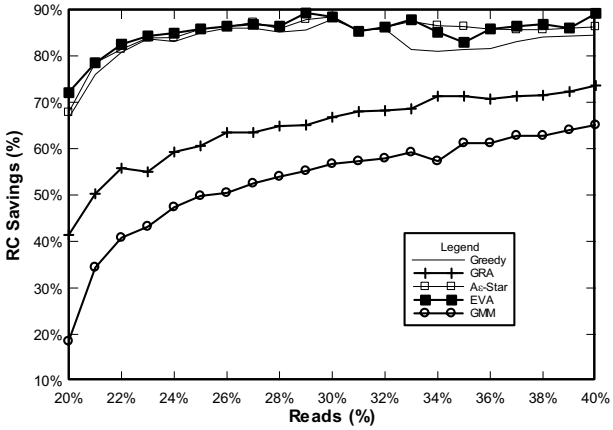

Fig. 4. RC Savings Versus Reads (N=2000, M=500, C=45%).

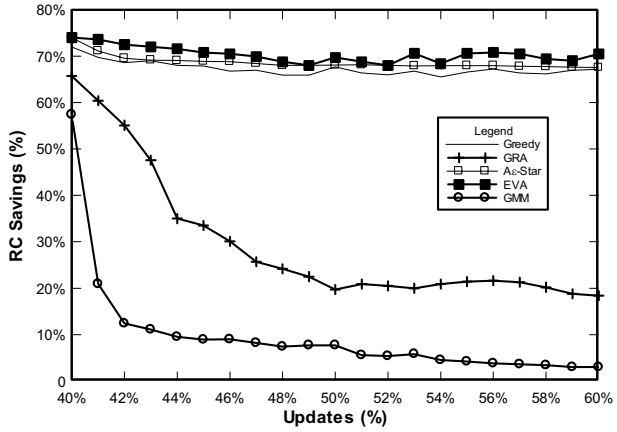

Fig. 5. RC Savings Versus Updates (N=2000, M=500, C=60%).

savings as only a small portion of the servers will be affected overall. Moreover, since objects are not equally read intensive, increase in the storage capacity would have a great impact at the beginning (initial increase in capacity), but has little effect after a certain point, where the most beneficial ones are already replicated. This is observable in Fig. 3, which shows the performance of the algorithms. Greedy and EVA showed an immediate initial increase (the point after which further replicating objects is inefficient) in its RC savings, but afterward showed a near constant performance. GMM and GRA although performed the worst, but observably gained the most RC savings (27% and 35%, respectively) followed by Greedy with 24%. Further experiments with various update ratios (5%, 10%, and 20%) showed similar plot trends. It is also noteworthy (plots not shown in this paper due to space restrictions) that the increase in capacity from 10% to 17%, resulted in 4 times (on average) more replicas for all the algorithms.

Next, we observe the effects of increase in the read and update (write) frequencies. Since these two parameters are

TABLE IV
AVERAGE RC SAVINGS IN PERCENTAGE

Problem Size	Greedy	GRA	Aε-Star	EVA	GMM
N=150, *M*=20 [*C*=20%,*U*=25%]	70.46	69.74	74.62	**75.70**	64.21
N=200, *M*=50 [*C*=20%,*U*=20%]	73.94	70.18	77.42	**78.43**	66.62
N=300, *M*=50 [*C*=25%,*U*=5%]	70.01	64.29	70.33	**82.25**	61.01
N=300, *M*=60 [*C*=35%,*U*=5%]	71.66	65.94	72.01	**74.43**	60.95
N=400, *M*=100 [*C*=25%,*U*=25%]	67.40	62.07	71.26	**73.89**	59.21
N=500, *M*=100 [*C*=30%,*U*=35%]	66.15	61.62	71.50	**75.45**	54.56
N=800, *M*=200 [*C*=25%,*U*=15%]	67.46	65.91	70.15	**73.68**	60.52
N=1000, *M*=300 [*C*=25%,*U*=35%]	69.10	64.08	70.01	**72.45**	61.16
N=1500, *M*=400 [*C*=35%,*U*=50%]	70.59	63.49	70.51	**74.01**	62.63
N=2000, *M*=500 [*C*=10%,*U*=60%]	67.03	63.37	72.16	**73.15**	60.94

complementary to each other, we describe them together. In both the setups the number of sites and objects were kept constant. Increase in the number of reads in the system would mean that there is a need to replicate as many object as possible (closer to the users). However, the increase in the number of updates in the system requires the replicas be placed as close as to the primary site as possible (to reduce the update broadcast). This phenomenon is also interrelated with the system capacity, as the update ratio sets an upper bound on the possible traffic reduction through replication. Thus, if we consider a system with unlimited capacity, the "replicate everywhere anything" policy is strictly inadequate. The read and update parameters indeed help in drawing a line between good and marginal algorithms. The plots in Figs. 4 and 5 show the results of read and update frequencies, respectively. A clear classification can be made between the algorithms. Aε-Star, Greedy and EVA incorporate the increase in the number of reads by replicating more objects and thus savings increase up to 89%. GMM gained the least of the RC savings of up to 54%. To understand why there is such a gap in the performance between the algorithms, we recall from [1] that GMM maintains a localized network perception. Increase in updates result in objects having decreased local significance (unless the vicinity is in close proximity to the primary location). On the other hand, Aε-Star, Greedy and EVA never

tend to deviate from their global view of the problem domain.

1) Summary

In summary, Table IV shows the quality of the solution in terms of RC percentage for 10 problem instances (randomly chosen), each being a combination of various numbers of sites and objects, with varying storage capacity and update ratio. For each row, the best result is indicated in bold. The proposed EVA steals the show in the context of solution quality, but Aε-Star and Greedy do indeed give a good competition, with savings within a range of 7%-10% of EVA.

VI. Conclusions

Manual mirroring of data objects is a tedious and time consuming operation. This paper proposed a game theoretical extended Vickrey auction (EVA) mechanism for object based data replication in large-scale distributed computing systems, such as, the Internet. EVA is a protocol for automatic replication and migration of objects in response to demand changes. EVA aims to place objects in the proximity of a majority of requests while ensuring that no hosts become overloaded.

EVA allows agents to compete for the scarce memory space at sites so that they can acquire the rights to place replicas. To cater for the possibility of cartel type behavior of the agents, EVA uses the extended Vickrey auction protocol. This leaves the agents with no option, then to report truthful valuations of the objects that they represent.

EVA was compared against some well-known techniques, such as: greedy, branch and bound and genetic algorithms. To provide a fair comparison, the assumptions and system parameters were kept the same in all the approaches. The experimental setup was designed to mimic a large-scale distributed computing system (the Internet), by using several Internet topology generators and World Cup Soccer 1998 web server access logs. The experimental results revealed that EVA outperformed the four widely cited and powerful techniques in both the execution time and solution quality. In summary, EVA exhibited 7%-10% better solution quality and 10%-30% savings in the algorithm termination timings.

References

[1] T. Loukopoulos, and I. Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *J. of Parallel and Distributed Comput.*, vol. 64, no. 11, pp. 1270-1285, 2004.

[2] B. Awerbuch, Y. Bartal and A. Fiat, "Competitive distributed file allocation," in *Proc. of 25th ACM Symp. on Theory Of Comput.,* Victoria, B.C., Canada, 1993, pp. 164-173.

[3] T. Abdelzaher and N. Bhatti, "Web content adaptation to improve sever workload behavior," *Comput. Networks*, vol. 21, no. 11, pp. 1536-1577, 1999.

[4] A. Heddaya and S. Mirdad, "WebWave: Globally load balanced fully distributed caching of hot published documents," *in Proc. 17th Intl. Conf. on Distributed Comput. Systems*, Baltimore, Maryland, 1997, pp. 160-168.

[5] J. Kangasharju, J. Roberts and K. Ross, "Object replication strategies in content distribution networks," in *Proc. of Workshop on Content Caching and Distribution*, 2001, pp. 455-466.

[6] S. Khan and I. Ahmad, "Heuristic-based replication schemas for fast information retrieval over the internet," in *Proc. of 17th Intl. Conf. on Parallel and Distributed Comput. Systems*, 2004, pp. 278-283.

[7] S. Mahmoud and J. Riordon, "Optimal allocation of resources in distributed information networks," *ACM Trans. on Database Systems*, vol. 1, no. 1, pp. 66-78, 1976.

[8] T. Loukopoulos, D. Papadias, and I. Ahmad, "An overview of data replication on the internet," in *Proc. of IEEE Intl. Symp. on Parallel Architectures, Algorithms and Networks,* 2002, pp. 31-36.

[9] W. Vickrey, "Counterspeculations, auctions and competitive sealed-bid tenders," *J. of Finance*, vol. 16, pp. 15-27, 1961.

[10] L. Qiu, V. Padmanabhan and G. Voelker, "On the placement of web server replicas," in *Proc. of the IEEE INFOCOM*, 2001, pp. 1587-1596.

[11] W. Chu, "Optimal file allocation in a multiple computer system," *IEEE Trans. on Computers*, vol. 18, no. 10, pp. 885-889, 1969.

[12] R. Casey, "Allocation of copies of a file in an information network," in *Proc. Spring Joint Computer Conf.*, IFIPS, 1972, pp. 617-625.

[13] K. Eswaran, "Placement of records in a file and file Allocation in a computer network," in *Proc. of Intl. Information Processing Conf.*, 1974, pp. 304-307.

[14] L. Dowdy and D. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, 1982.

[15] K. Chandy and J. Hewes, "File allocation in distributed systems," in *Proc. of the International Symp. on Comput. Performance Modeling, Measurement and Evaluation*, 1976, pp. 10-13.

[16] S. Hakimi, "Optimum location of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, pp. 450-459, 1964.

[17] S. Jamin, C. Jin, Y. Jin, D. Riaz, Y. Shavitt and L. Zhang, "On the placement of internet instrumentation," in *Proc. of the IEEE INFOCOM*, 2000, pp. 295-304.

[18] M. Karlsson and M. Mahalingam, "Do we need replica placement algorithms in content delivery networks?" in *Proc. of Web Caching and Content Distribution Workshop*, 2002, pp. 117-128.

[19] S. Cook, J. Pachl, and I. Pressman, "The optimal location of replicas in a network using a READ-ONE-WRITE-ALL policy," *Distributed Computing*, vol. 15, no. 1, pp. 57-66, 2002.

[20] S. Khan and I. Ahmad, "A powerful direct mechanism for optimal www content replication," in *Proc. of 19th IEEE International Parallel and Distributed Processing Symposium*, 2005, p. 86.

[21] S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, "Constrained mirror placement on the internet," in *Proc. of the IEEE INFOCOM*, 2001, pp. 31-40.

[22] B. Li, M. Golin, G. Italiano and X. Deng, "On the optimal placement of web proxies in the internet," in *Proc. of the IEEE INFOCOM*, 2000, pp. 1282-1290.

[23] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal placement of replicas in trees with read, write, and storage Costs," *IEEE Trans. on Parallel and Distributed Systems*, vol. 12, no. 6, pp. 628-637, 2001.

[24] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," in *Proc. of IEEE INFOCOM*, 2001, pp. 1773-1780.

[25] P. Krishnan, D. Raz, and Y. Shavitt, "The Cache Location Problem," *IEEE/ACM Trans. on Networking*, 8(5), pp. 568-582, 2000.

[26] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384-392, 2002.

[27] A. Venkataramanj, P. Weidmann, and M. Dahlin, "Bandwidth constrained placement in a WAN," in *Proc. ACM Symp. on Principles of Distributed Computing*, 2001, pp. 134-143.

[28] M. Korupolu and C. Plaxton, "Analysis of a local search heuristic for facility location problems," *J. of Algorithms*, vol. 37, no. 1, pp. 146-188, 2000.

[29] C. Krick, H. Racke, and M. Westermann, "Approximation algorithms for data management in networks," in *Proc. of the Symp. on Parallel Algorithms and Architecture*, 2001, pp. 237-246.

[30] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou and J. Kubiatowicz, "Selfish caching in distributed systems: A game-theoretic analysis," in *Proc. of 23rd ACM Symp. on Principles of Distributed Computing*, 2004, pp. 21-30.

[31] N. Laoutaris, O. Telelis, V. Zissimopoulos and I. Stavrakakis, "Local utility aware content replication," in *IFIP Networking Conference*, 2005, pp. 455-468.

[32] B. Narebdran, S. Rangarajan and S. Yajnik, "Data distribution algorithms for load balancing fault-tolerant web access," in *Proc. of the 16th Symp. on Reliable Distributed Systems,* 1997, pp. 97-106.

[33] M. Rabinovich, "Issues in web content replication," *Data Engineering Bulletin,* vol. 21, no. 4, pp. 21-29, 1998.

[34] M. Arlitt and T. Jin, "Workload characterization of the 1998 World Cup Web Site," tech. report, HP Lab, Palo Alto, HPL-1999-35(R.1), 1999.
[35] K. Calvert, M. Doar, E. Zegura, "Modeling Internet topology," *IEEE Communications*, 35(6), pp. 160-163, 1997.
[36] P. Apers, "Data Allocation in Distributed Database Systems," *ACM Trans. Database Systems*, 13(3), pp. 263-304, 1988.

Samee Ullah Khan (M'05) received the B.S. degree in computer science and engineering from Ghulam Ishaq Khan Institute of Engineering Science and Technology, Topi, Pakistan in 1999, and became a member of the International Enformatika Society (IES) in 2005.

He is currently a graduate student in the Computer Science and Engineering Department of the University of Texas at Arlington, TX, USA. His research interests include algorithmic mechanism design, game theoretical applications, combinatorial games, operations research, combinatorial optimization, and distributed computing algorithms.

Mr. Khan is a member of the European Association of Theoretical Computer Science, the Game Theory Society, the IEEE Communications Society, the IEEE Computer Society, and the Society of Photo-Optical Instrumentation Engineers. He also serves on the IES scientific committee.

Ishfaq Ahmad received the B.Sc. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985, the M.S. degree in computer engineering, and the Ph.D. degree in computer science, both from Syracuse University, Syracuse, NY, in 1987 and 1992, respectively.

He is currently a Full Professor of Computer Science and engineering in the Computer Science and Engineering Department, University of Texas (UT) at Arlington. Prior to joining UT Arlington, he was an associate professor in the Computer Science Department at Hong Kong University of Science and Technology (HKUST), Hong Kong. At HKUST, he was also the Director of the Multimedia Technology Research Center, an officially recognized research center that he conceived and built from scratch. The center was funded by various agencies of the Government of the Hong Kong Special Administrative Region as well as local and international industries. With more than 40 personnel including faculty members, postdoctoral fellows, full-time staff, and graduate students, the center engaged in numerous research and development projects with academia and industry from Hong Kong, China, and the U.S. Particular areas of focus in the center are video (and related audio) compression technologies, video telephone and conferencing systems. The center commercialized several of its technologies to its industrial partners world wide. His recent research focus has been on developing parallel programming tools, scheduling and mapping algorithms for scalable architectures, heterogeneous computing systems, distributed multimedia systems, video compression techniques, and web management. His research work in these areas is published in over 150 technical papers in refereed journals and conferences.

Dr. Ahmad has received Best Paper Awards at Supercomputing'90 (New York), Supercomputing'91 (Albuquerque), and the 2001 International Conference Parallel Processing (Spain). He has participated in the organization of several international conferences and is an Associate Editor of Cluster Computing, Journal of Parallel and Distributed Computing, IEEE Transactions on Circuits and Systems for Video Technology, IEEE Concurrency, and IEEE Distributed Systems Online.