International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:5, 2012

A Fast Replica Placement Methodology for Large-scale Distributed Computing Systems

Samee Ullah Khan and Cemal Ardil

***Abstract*—**Fine-grained data replication over the Internet allows duplication of frequently accessed data objects, as opposed to entire sites, to certain locations so as to improve the performance of large-scale content distribution systems. In a distributed system, agents representing their sites try to maximize their own benefit since they are driven by different goals such as to minimize their communication costs, latency, etc. In this paper, we will use game theoretical techniques and in particular auctions to identify a bidding mechanism that encapsulates the selfishness of the agents, while having a controlling hand over them. In essence, the proposed game theory based mechanism is the study of what happens when independent agents act selfishly and how to control them to maximize the overall performance. A bidding mechanism asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals. Experimental results reveal that this mechanism provides excellent solution quality, while maintaining fast execution time. The comparisons are recorded against some well known techniques such as greedy, branch and bound, game theoretical auctions and genetic algorithms.

***Keywords*—**Data replication, auctions, static allocation, pricing.

I. Introduction

In the Internet a magnitude of heterogeneous entities (e.g. providers and commercial services) offer, use, and even compete with each other for resources. The Internet is emerging as a new platform for distributed computing and brings with it problems never seen before. New solutions should take into account the various new concepts derived from multi-agent systems in which the agents cannot be assumed to act in accordance to the deployed algorithm. In a heterogeneous system such as the Internet entitles act selfishly. This is obvious since they are driven by different goals such as to minimize their communication costs, latency, etc. Thus, one cannot assume that agents would follow the protocol or the algorithm, though they respond to incentives (e.g. payments received for compensation).

In this paper, we will use game theoretical techniques and in particular auctions to identify a bidding mechanism that encapsulates the selfishness of the agents, while having a controlling hand over them. This work is inspired from the work reported in [11] and [14]. In essence, game theory is the study of what happens when independent agents act selfishly. A bidding mechanism asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals.

In this paper, we will apply the derived mechanism to the fine grained data replication problem (DRP) over the Internet. This problem strongly conforms to the selfish agents' notion and has a wide range of applications. Replication is widely used to improve the performance of large-scale content distribution systems such as the CDNs [13]. Replicating the data over geographically dispersed locations reduces access latency, network traffic, and in turn adds reliability, robustness and fault-tolerance to the system. Discussions in [4], [5], [8], [9], and [12] reveal that client(s) experience reduced access latencies provided that data is replicated within their close proximity. However, this is applicable in cases when only read accesses are considered. If updates of the contents are also under focus, then the locations of the replicas have to be: 1) in close proximity to the client(s), and 2) in close proximity to the primary (assuming a broadcast update model) copy. For fault-tolerant and highly dependable systems, replication is essential, as demonstrated in a real world example of OceanStore [13]. Therefore, efficient and effective replication schemas strongly depend on how many replicas to be placed in the system, and more importantly where. Needless to say that our work differs form the existing techniques in the usage of game theoretical techniques. To the best of the authors' knowledge this is the very first work that addresses the problem using such techniques.

The remainder of this paper is organized as follows. Section II formulates the DRP. Section III concentrates on modeling the resource allocation mechanism for the DRP. The experimental results and concluding remarks are provided in Sections IV and V, respectively.

II. 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 $1 \leq i \leq M$. 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

S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58102, USA (phone: 701-231-7615; fax: 701-231-8677; e-mail: samee.khan@ ndsu.edu).

C. Ardil is with the National Academy of Aviation, Baku, Azerbaijan, (e-mail: cemalardil@gmail.com).

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], [9], and [10]). Let there be N objects, each identifiable by a unique name O_k and size in simple data unites o_k where $1 \leq k \leq N$. Let r_k^i and w_k^i be the total number of reads and writes, respectively, initiated from S_i for O_k.

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 DRP under consideration, we are interested in minimizing the total Object Transfer Cost (OTC) 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 OTC. The first component of OTC is due to the read requests. Let R_k^i denote the total OTC, 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 OTC is the cost arising due to the writes. Let W_k^i be the total OTC, 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:

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

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 OTC, denoted as $C_{overall}$, due to reads and writes is given by:

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

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:

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

Sites which are not the *replicators* of object O_k create OTC 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 DRP 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*:

$\sum_{k=1}^{N} X_{ik} o_k \leq s_i \quad \forall (1 \leq i \leq M)$,

and subject to the primary copies policy:

$X_{P_k k} = 1 \quad \forall (1 \leq k \leq N)$."

In the generalized case, the DRP is NP-complete [9].

III. Our Proposed Procedure

We follow the same pattern as discussed in Section II.

The Basics: The distributed system described in Section III is considered, where each site is represented by an agent, *i.e.*, the mechanism contains M agents. In the context of the DRP, an agent holds two key elements of data a) the available site capacity ac^i, and b) the cost to replicate ($RC_k^i = R_k^i + W_k^i$) an object k to the agent's site i..

Intuitively, if agents know the available site capacities of other agents, that gives them no advantage whatsoever. However, if they come about to know their replication cost then they can modify their valuations and alter the algorithmic output. It is to be noted that an agent can only calculate the replication cost via the frequencies of reads and writes. Everything else such as the network topology, latency on communication lines, and even the site capacities can be public knowledge. Therefore, DRP[π] is the only natural choice. A rigorous proof on the validity of DRP[π] is presented in [7].

Communications: The agents in the mechanism are assumed to be selfish and therefore, they project a bid b^i to the mechanism. In reality the amount of communications made are immense. This fact was not realized in [3], where the authors assume superfluous assumptions on the implementation. In the later text we will reveal how to cope with this dilemma.

Components: The mechanism has two components 1) the algorithmic output $x(\cdot)$, and 2) the payment mapping function $p(\cdot)$.

Algorithmic output: In the context of the DRP, the algorithm accepts bids from all the agents, and outputs the maximum beneficial bid, *i.e.*, the bid that incurs the minimum replication cost overall (Equation 3). We will give a detailed description of the algorithm in the subsequent text.

Monetary cost: When an object is allocated (for replication) to an agent *i*, the agent becomes responsible to entertain (read and write) requests to that object. For example, assume object *k* is replicated to agent *i*. Then the amount of traffic that the agent has to entertain due to the replication of object *k* is exactly equivalent to the replication cost, i.e., $c^i = RC_k^i$. This fact is easily deducible from Equation 4.

Payments: To offset c^i, the mechanism makes a payment $p^i(b)$ to agent *i*. This payment is equivalent to the cost it incurs to replicate the object, i.e., $p^i(b) = c^i$. The readers would immediately note that in such a payment agent *i* can never get a profit greater than 0. This is exactly what we want. In a selfish environment, it is possible that the agents bid higher than the true value, the mechanism creates an illusion to negate that. By compensating the agents with the exact amount of what the cost occurs, it leaves no room for the agents to overbid or underbid (in the later text we will rigorously prove the above argument). Therefore, the voluntary characteristic of the mechanism now becomes a strongly voluntary and we quote the following definition.

Definition 1: A mechanism is characterized as a strongly voluntary participation mechanism if for every agent i, $u^i(t^i,(b^{-i},t^i)) = 0$ [14].

We want to emphasis that each agent's incentive is to replicate objects so that queries can be answered locally, for the sake of users that access the agent's site. If the replicas are made available elsewhere, the agent may lose the users, as they might divert their accesses to other sites.

Bids: Each agent *i* reports a bid that is the direct representation of the true data that it holds. Therefore, a bid b^i is equivalent to $1/RC_k^i$. That is, the lower the replication cost the higher is the bid and the higher are the chances for the bid b^i to win.

In essence, the mechanism $m(x(b),p(b))$, takes in the vector of bids *b* from all the agents, and selects the highest bid. The highest bidder is allocated the object *k* which is added to its allocation set x^i. The mechanism then pays the bidder p^i. This payment is equivalent to the cost incurred due to entertain requests from object *k* by users. The pseudo-code of the mechanism is given in Fig. 1.

Description of Algorithm: We maintain a list L^i at each server. This list contains all the objects that can be replicated by agent *i* onto site S^i. We can obtain this list by examining the two constraints of the DRP. List L^i would contain all the objects that have their size less then the total available space ac^i. Moreover, if site S^i is the primary host of some object *k*', then *k*' should not be in L^i. We also maintain a list *LS* containing all sites that can replicate an object, *i.e.*, $S^i \in LS$ if $L^i \neq$NULL. The algorithm works iteratively. In each step the mechanism asks all the agents to send their preferences (first PARFOR loop). Each agent *i* recursively calculates the true data of every object in list L^i. Each agent then reports the dominant true data (line 09) to the mechanism. The mechanism receives all the corresponding entries, and then chooses the globally dominant true data. This is broadcasted to all the agents, so that they can update their nearest neighbor table NN_k^i, which is shown in Line 20 (NN^i_{OMAX}). The object is replicated and the payment is made to the agent. The mechanism progresses forward till there are no more agents interested in acquiring any data for replication.

Now, we present some results that strengthen our claim on the optimality of the derived bidding mechanism. We begin by making the following observations.

Assume that the mechanism $m = (x(b),p(b))$ is truthful and each payment $p^i(b^{-i},b^i)$ and allocation $x^i(b^{-i},b^i)$ is twice differentiable with respect to b^i, for all the values of b^{-i}. We fix some agent *i* and derive a formula for p^i, allocation x^i, and profit to be the functions of just agent *i*'s bid b^i. Since agent *i*'s profit is always maximized by bidding truthfully (Lemma 1), the derivative is zero and the second derivative is non-positive at t^i. Since this holds no matter what the value of t^i is, we can integrate to obtain an expression for p^i. We state: $p^i(b^i) = p^i(0)+b^ix^i(b^i)-\int_0^{bi}x^i(u)du$. This is now the basis of our extended theoretical results. Literature survey revealed the following two important characteristics of a frugal payment mechanism. We state them below.

Definition 2: With the other agents' bid b^{-i} *fixed, consider* $x^i(b^{-i},b^i)$ *as a single variable function of* b^i. *We call this the allocation curve or the allocation profile of agent i. We say the output function x is decreasing if each of the associated allocation curves is decreasing, i.e.,* $x^i(b^{-i},b^i)$ *is a decreasing function of* b^i*, for all i and* b^{-i}.

Based on Definition 2, we can state the following.

Theorem 1: A mechanism is truthful if its output function $x(b)$ *is decreasing.*

Proof: We prove this for the DRP mechanism. For simplicity we fix all bids b^{-i}, and focus on $x^i(b^{-i},b^i)$ as a single variable function of b^i, *i.e.*, the allocation x^i would only change if b^i is altered. We now consider two bids b^i and $b^{i'}$ such that $b^{i'} > b^i$. In terms of the true data t^i, this conforms to $RC_k^{i'} > RC_k^i$. Let x^i and $x^{i'}$ be the allocations made to the agent *i* when it bids b^i and $b^{i'}$, respectively. For a given allocation, the total replication cost associated can be represented as $C^i=\sum_{k\in xi}RC_k^i$. The proof of the theorem reduces to proving that $x^{i'} < x^i$, *i.e.*, the allocation computed by the algorithmic output is decreasing in b^i. The proof is simple by contradiction. Assume that that $x^{i'} \geq x^i$. This implies that $1/(C^i-RC_k^i) < 1/(C^{i'}-RC_k^i) \leq 1/(C^i-RC_k^{i'})$. This means that there must be an agent *-i* who has a bid that

```
The Mechanism (FA)

Initialize:
01 $LS$, $L^i$, $T_k^i$, $M$, $MT$
02 WHILE $LS \neq$ NULL DO
03     $OMAX$ = NULL; $MT$ = NULL; $P^i$ = NULL;
```

```
    PARFOR each $S^i \in$ LS DO
        FOR each $O_k \in L^i$ DO
            $T_k^i$ = compute ($t^i$);  /*compute the valuation corresponding to the desired object*/
        ENDFOR
      $t^i$ = argmax$_k$($T_k^i$);
      SEND $t^i$ to M; RECEIVE at $M$ $t^i$ in $MT$;
    ENDPARFOR
OMAX = argmax$_k$($MT$);    /*Choose the global dominate valuation*/
$P^i$ = 1/OMAX;              /*Calculate the payment*/
BROADCAST $OMAX$;
SEND $P^i$ to $S^i$;         /*Send payments to the agent who is allocate the object OMAX*/
Replicate $O_{OMAX}$;
$ac^i$=$ac^i$ - $o_k$;                     /*Update capacity*/
$L^i$ = $L^i$ - $O_k$;               /*Update the list*/
IF $L^i$ = NULL THEN SEND info to $M$ to update $LS$ = $LS$ - $S^i$;        /*Update mechanism players*/
    PARFOR each $S^i \in$ LS DO
        Update $NN^i_{OMAX}$          /*Update the nearest neighbor list*/
    ENDPARFOR               /*Get ready for the next round*/
ENDWHILE
```

Fig. 1 Frugal Auction (FA) Mechanism

supersedes b^{i}'. But that is not possible as we began with the assumption that all other bids are fixed so there can be no other agent -i. If i = -i, then that is also not possible since we assumed that b^{i}' > b^i. ■

The result in Theorem 1 is extended as:

Theorem 2: A *decreasing output function admits a truthful payment scheme satisfying voluntary participation if and only if* $\int_0^\infty x^i(b^{-i}, u)du < \infty$ *for all i,* b^{-i}. *In this case we can take the payments to be:*

$$p^i(b^{-i}, b^i) = b^i x^i(b^{-i}, b^i) + \int_0^\infty x^i(b^{-i}, u)du \ .$$

Proof: The first term $b^i x^i(b^{-i}, b^i)$ compensates the cost incurred by agent i to host the allocation x^i. The second term $\int_0^\infty x^i(b^{-i}, u)du$ represents the expected profit of agent i. If agent i bids its true value t^i, then its profit is

$$= u^i(t^i, (b^{-i}, t^i))$$

$$= t^i x^i(b^{-i}, t^i) + \int_{t^i}^\infty x^i(b^{-i}, x)dx - t^i x^i(b^{-i}, t^i)$$

$$= \int_{t^i}^\infty x^i(b^{-i}, x)dx$$

If agent i bids its true value, then the expected profit is greater than in the case it bids other values. We explain this as follows: If agent i bids higher (b^{i}'>t^i), then the expected profit is

$$= u^i(t^i, (b^{-i}, b^{i}'))$$

$$= b^i{}' x^i(b^{-i}, b^i{}') + \int_{b^i{}'}^\infty x^i(b^{-i}, x)dx - t^i x^i(b^{-i}, b^i{}')$$

$$= (b^i{}' - t^i) x^i(b^{-i}, b^i{}') + \int_{b^i{}'}^\infty x^i(b^{-i}, x)dx \ .$$

Because $\int_{b^i{}'}^\infty x^i(b^{-i}, x)dx < \infty$ and $b^{i}{}' > t^i$, we can express the profit when agent i bids the true value as follows:

$$\int_{t^i}^{b^i{}'} x^i(b^{-i}, x)dx + \int_{b^i{}'}^\infty x^i(b^{-i}, x)dx \ .$$

This is because x^i is decreasing in b^i and b^{i}' > t^i, we have the following equation:

$$(b^i{}' - t^i) x^i(b^{-i}, b^i{}') < \int_{t^i}^{b^i{}'} x^i(b^{-i}, x)dx \ .$$

From this relation, it can be seen that the profit with overbidding is lower then the profit with bidding the true data. Similar arguments can be used for underbidding. ■

IV. Experimental Results

We performed experiments on a 440MHz Ultra 10 machine with 512MB memory. The experimental evaluations were targeted to benchmark the placement policies. The resource allocation mechanism was implemented using IBM Pthreads.

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 (nodes). We used existing topology generator toolkits and also self generated networks. In all the topologies the distance of the link between nodes was equivalent to the communication cost. Table I summarizes the various techniques used to gather forty-five various topologies for networks with 100 nodes.

To evaluate the chosen replication placement techniques on realistic traffic patterns, we used the access logs collected at

Table I
Parameter interval variance characterization for topologies with 100 nodes

Topology	Mathematical Representation	Parameter Interval Variance
SGRG (12 topologies)	Randomized layout with node degree (d*) and Euclidian distance (d) between nodes as parameters.	d={5,10,15,20}, d*={10,15,20}.

GT-ITM PR [4] (5 topologies)	Randomized layout with edges added between the randomly located vertices with a probability (p).	p={0.4,0.5,0.6,0.7,0.8}.
GT-ITM W [4] (9 topologies)	$P(u,v)=\alpha e^{-d/(\beta L)}$	α={0.1,0.15,0.2,0.25}, β={0.2,0.3,0.4}.
SGFCGUD (5 topologies)	Fully connected graph with uniform link distances (d).	d_1=[1,10],d_2=[1,20],d_3=[1,50], d_4=[10,20], d_5=[20,50].
SGFCGRD (5 topologies)	Fully connected graph with random link distances (d).	d_1=[1,10],d_2=[1,20],d_3=[1,50], d_4=[10,20], d_5=[20,50].

SGRGLND
(9 topologies)
	Random layout with link distance having a lognormal distribution [9].	μ={8.455,9.345,9.564},
		σ={1.278,1.305,1.378}.

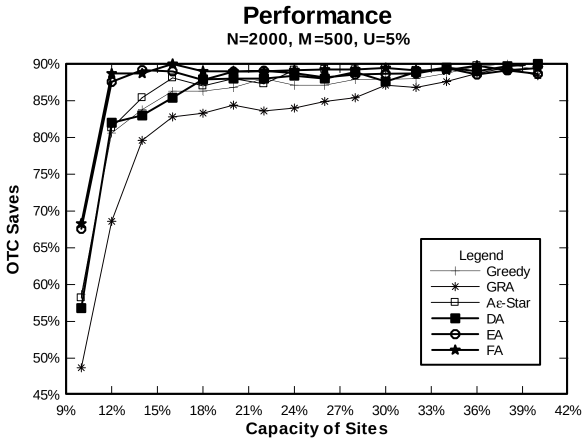

Fig. 2 OTC savings versus capacity

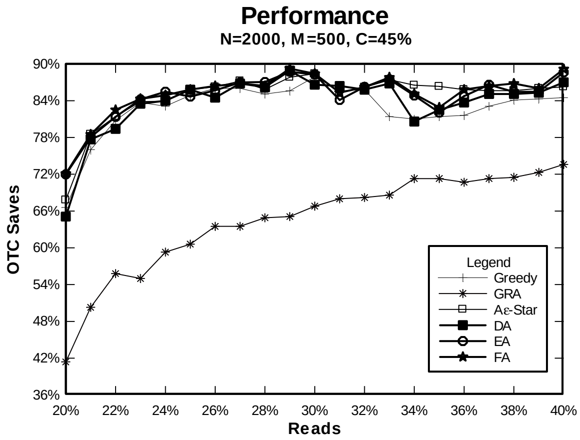

Fig. 3 OTC savings versus reads

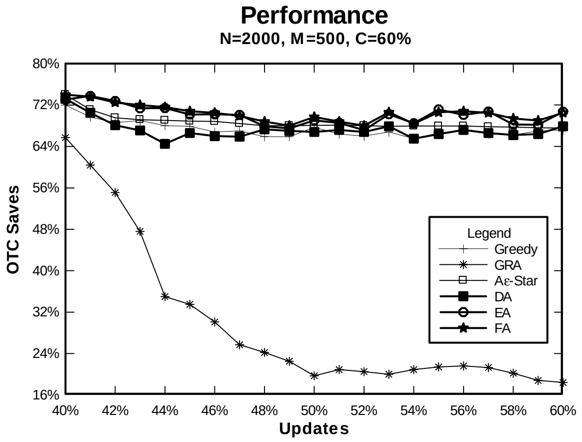

Fig. 4 OTC savings versus updates

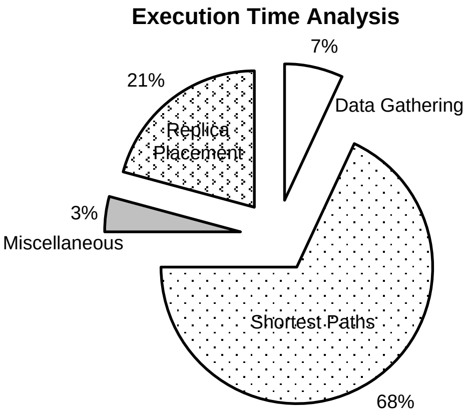

Fig. 5: Execution time components.

the Soccer World Cup 1998 website [1]. Each experimental setup was evaluated thirteen times, *i.e.*, only the 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 chose the top five hundred clients (maximum experimental setup). A random mapping was then performed of the clients to 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

TABLE II
RUNNING TIME IN SECONDS [C=45%, U=15%]

Problem Size	Greedy	GRA	Aε-Star	DA	EA	FA
M=300, N=1350	190.01	242.12	247.66	**87.92**	164.15	93.26

M=300, *N*=1400	206.26	326.82	279.45	**95.64**	178.90	97.98
M=300, *N*=1450	236.61	379.01	310.12	115.19	185.15	**113.65**
M=300, *N*=1500	258.45	409.17	333.03	127.10	191.24	**124.73**
M=300, *N*=1550	275.63	469.38	368.89	**143.94**	197.93	147.16
M=300, *N*=2000	298.12	475.02	387.94	**158.45**	204.29	159.12
M=400, *N*=1350	321.60	492.10	353.08	**176.51**	218.15	176.90
M=400, *N*=1400	348.53	536.96	368.03	**187.26**	223.56	195.41
M=400, *N*=1450	366.38	541.12	396.96	**192.41**	221.10	214.55
M=400, *N*=1500	376.85	559.74	412.17	**208.92**	245.47	218.73
M=400, *N*=1550	389.71	605.63	415.55	**215.24**	269.31	223.92
M=400, *N*=2000	391.55	659.39	447.97	**224.18**	274.24	235.17
M=500, *N*=1350	402.20	660.86	460.44	**246.43**	284.63	259.56
M=500, *N*=1400	478.10	689.44	511.69	**257.96**	301.72	266.42
M=500, *N*=1450	485.34	705.07	582.71	**269.45**	315.13	272.68
M=500, *N*=1500	511.06	736.43	628.23	**278.15**	324.26	291.83
M=500, *N*=1550	525.33	753.50	645.26	**289.64**	331.57	304.47
M=500, *N*=2000	539.15	776.99	735.36	**312.68**	345.94	317.60

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.

For comparison, we selected five various types of replica placement techniques. To provide a fair comparison, the assumptions and system parameters were kept the same in all the approaches. The techniques studied include efficient branch-and-bound based technique (Aε-Star [5]). For fine-grained replication, the algorithms proposed in [6], [8], [9], and [12] are the only ones that address the problem domain similar to ours. We select from [12] the greedy approach (Greedy) for comparison because it is shown to be the best compared with 4 other approaches (including the proposed technique in [8]); thus, we indirectly compare with 4 additional approaches as well. Algorithms reported in [6] (Dutch (DA) and English auctions (EA)) and [9] (Genetic based algorithm (GRA)) are also among the chosen techniques for comparisons. Due to space limitations we will only give a brief overview of the comparative techniques. Details for a specific technique can be obtained from the referenced papers.

The solution quality is measured in terms of network communication cost (OTC percentage) that is saved under the replication scheme found by the algorithms, compared to the initial one, *i.e.*, when only primary copies exists.

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 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. 2, which shows the performance of the algorithms. GRA once again performed the worst. The gap between all other approaches was reduced to within 7% of each other. DA and FA showed an immediate initial increase (the point after which further replicating objects is inefficient) in its OTC savings, but afterward showed a near constant performance. GRA although performed the worst, but observably gained the most OTC savings (35%) followed by Greedy with 29%. 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 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. 3 and 4 show the results of read and update frequencies, respectively. A clear classification can be made between the algorithms. Aε-Star, DA, EA, Greedy and FA incorporate the increase in the number of reads by replicating more objects and thus savings increase up to 88%. GRA gained the least of the OTC savings of up to 67%. To understand why there is such a gap in the performance between the algorithms, we should recall that GRA specifically depend on the initial population (for details see [9]). Moreover, GRA 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, DA, EA, Greedy and FA never tend to deviate from their global view of the problem search space.

Lastly, we compare the termination time of the algorithms. Before we proceed, we would like to clarify our measurement of algorithm termination timings. The approach we took was to see if these algorithms can be used in dynamic scenarios. Thus, we gather and process data as if it was a dynamic system. The average breakdown of the execution time of all the algorithms combined is depicted in Fig. 5. There 68% of all the algorithm termination time was taken by the repeated calculations of the shortest paths. Data gathering and dispersion, such as reading the access frequencies from the processed log, etc. took 7% of the total time. Other miscellaneous operations including I/O were recorded to carry 3% of the total execution time. From the plot it is clear that a totally static setup would take no less that 21% of the time depicted in Table II.

Various problem instances were recorded with *C*=45% and *U*=15%. Each problem instance represents the average recorded time over all the 45 topologies and 13 various access logs. The entries in bold represent the fastest time recorded over the problem instance. It is observable that FA and DA terminated faster than all the other techniques, followed by EA, Greedy, Aε-Star and GRA. If a static environment was considered, FA with the maximum problem instance would have terminated approximately in 66.69 seconds (21% of the algorithm termination time).

In summary, based on the solution quality alone, the algorithms can be classified into four categories: 1) The very high performance algorithms that include EA and FA, 2) the high performance algorithms of Greedy and DA, 3) the medium-high performance Aε-Star, and finally 4) the mediocre performance algorithm of GRA. While considering the termination timings, FA and DA did extremely well, followed by EA, Greedy, Aε-Star, and GRA.

V. Conclusions

This paper proposed a game theoretical resource allocation mechanism that effectively addressed the fine-grained data replication problem with selfish players. The experimental results which were recorded against some well know techniques such as branch and bound, greedy, game theoretical auctions, and genetic algorithms revealed that the proposed mechanism exhibited 5%-10% better solution quality and incurred fast execution time.

References

[1] M. Arlitt and T. Jin, "Workload characterization of the 1998 World Cup Web Site," Technical Report, Hewlett Packard Lab, Palo Alto, HPL-1999-35(R.1), 1999.

[2] K. Calvert, M. Doar, E. Zegura, "Modeling Internet Topology," *IEEE Communications*, 35(6), pp. 160-163, 1997.

[3] D. Grosu and A. Chronopoulos, "Algorithmic Mechnism Design for Load Balancing in Distributed Systems," *IEEE Trans. Systems, Man and Cybernatics B*, 34(1), pp. 77-84, 2004.

[4] S. Jamin, C. Jin, Y. Jin, D. Riaz, Y. Shavitt and L. Zhang, "On the Placement of Internet Instrumentation," in *IEEE INFOCOM*, 2000.

[5] S. Khan and I. Ahmad, "Heuristic-based Replication Schemas for Fast Information Retrevial over the Internet," in *17th International Conference on Parallel and Distributed Computing Systems*, San Francisco, U.S.A., 2004.

[6] S. Khan and I. Ahmad, "A Game Theoretical Solution for Web Content Replication," Technical Report, CSE-UTA, 2004.

[7] S. Khan and I. Ahmad, "A Pure Nash Equilibrium-based Game Theoretical Method for Data Replication across Multiple Servers," IEEE Transactions on Knowledge and Data Engineering, accepted to appear in 2009.

[8] B. Li, M. Golin, G. Italiano and X. Deng, "On the Optimal Placement of Web Proxies in the Internet," in *IEEE INFOCOM*, 2000.

[9] T. Loukopoulos, and I. Ahmad, "Static and Adaptive Distributed Data Replication using Genetic Algorithms," *Journal of Parallel and Distributed Computing*, 64(11), pp. 1270-1285, 2005.

[10] T. Loukopoulos, I. Ahmad, and D. Papadias, "An Overview of Data Replication on the Internet," in *IEEE International Symposium on Parallel Processing and Networking*, pp. 31-36, 2002.

[11] N. Nisan and A. Ronen, "Algorithimic Mechanism Design," in *31st ACM Symposium on Theory of Computing*, pp. 129-140, 1999.

[12] L. Qiu, V. Padmanabhan and G. Voelker, "On the Placement of Web Server Replicas," in *IEEE INFOCOM*, 2001.

[13] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, J. Kubiatowicz, "Maintenance-free Global Storage," *IEEE Internet Computing*, 5(5), pp. 40-49, 2001.

[14] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," *Journal of Finance*, pp. 8-37, 1961.