A Semi-Distributed Axiomatic Game Theoretical Mechanism for Replicating Data Objects in Large Distributed Computing Systems

Samee Ullah Khan and Ishfaq Ahmad
Department of Computer Science and Engineering
University of Texas, Arlington, TX 76019, U.S.A.
{sakhan, iahmad}@cse.uta.edu

Abstract

Replicating data objects onto servers across a system can alleviate access delays. The selection of data objects and servers requires solving a constraint optimization problem, which is NP-complete in general. A majority of conventional replica placement techniques falter on issues of scalability or solution quality. To counteract such issues, we propose a game theoretical replica placement technique, in which computational agents compete for the allocation or reallocation of replicas onto their servers in order to reduce the user perceived access delays. The technique is based upon six well-defined axioms, each guaranteeing certain basic game theoretical properties. This eccentric method of designing game theoretical techniques using axioms is unique in the literature and takes away from the designers the cumbersome mathematical details of game theory. The distinctive feature of these axioms is that when amassed together, their individual properties constrict into one system-wide performance enhancement property, which in our case is the reduction of access time. The control of the proposed technique is "semi-distributed" in nature, wherein all the heavy processing is done on the servers of the distributed system and the central body is only required to take a binary decision: (0) not to replicate or (1) to replicate. This semi-distributed approach makes the technique scalable and helps solutions to converge in a fast turn-around time without loosing much of the solution quality. Experimental comparisons are made against: 1) branch and bound, 2) greedy, 3) genetic, 4) Dutch auction, and 5) English auction. As attested by the results, the proposed technique maintains superior solution quality in terms of lower communication cost and reduced execution time.

1. Introduction

Numerous methods have been proposed that address the data object replication problem in large distributed computing systems. (For a recent survey see [20].) However, all of the reported methods operate under the generic assumption that the servers cooperate with each other in order to attain system-wide benefits. For instance, in a content distribution network (CDN) the *distribution system* moves contents to the replica servers [12], [13], [14]. This distribution system acts as a centralized decision making body, and makes decisions on where and what to replicate by observing the system's parameters such as server load, storage capacity, communication latencies, etc. Although the Internet encourages distributed control, it requires that the overall system performance criteria be met. Managing a large distributed system, such as the Internet, through a single entity would require huge amounts of data processing in order to find a feasible replica schema and would be vulnerable to system failures [6]. Consequently, we propose a "semi-distributed" [1] (a hybrid of both the centralized and decentralized approaches) replication technique, by which all the heavy processing is done on the servers of the distributed system and the central body is only required to take a binary decision: (0) not to replicate or (1) to replicate. This leads to a simple yet an efficient scheme with enhanced scalability and low complexity that grows in proportion to the number of servers in the system.

For the proposed semi-distributed replica allocation technique, we abstract the Internet as an agent based model wherein agents continuously compete for allocation and reallocation of data objects. An agent is a computational entity that is capable of autonomous behaviour in the sense of being aware of the options available to it when faced with a decision making task related to its domain of interest [28]. In such a competitive model, there is no a-priori motivation for cooperation and the agents due to their localized view of the problem domain are encouraged to take decisions individually that are beneficial only to themselves. To cope with such individualism and localized optimization, new mechanisms based on novel oracles need to be derived. The objective of the mechanisms should be to allow the agents to take decisions that are based on their

1-4244-0910-1/07/$20.00 ©2007 IEEE.

local information, which essentially translate into the overall system performance enhancement.

This paper aims to formally specify a semi-distributed game theoretical replica allocation mechanism, in which autonomous agents compete to replicate data objects in a non-cooperative game. Briefly, the mechanism works as follows. It first asks all the agents to provide a list of objects that are beneficial for replication onto their servers. Having obtained this data, the mechanism makes the allocation and informs the agents. For every allocation the mechanism makes a payment to each agent (to compensate for hosting object(s)). Each agent's goal is to provide the mechanism with a list of objects that maximize its benefit. The mechanism on the other hand, is designed such that, the agents maximize their benefit only if the reported list contains objects that when replicated, brings the overall system communication cost to the minimum. Thus, the agents which are competing against each other in a non-cooperative game collaborate unknowingly to optimize the overall system goal.

The major contributions of this paper are as follows:

1. We derive a general purpose axiomatic game theoretical mechanism. This mechanism ensures that although, the agents use self-beneficial strategies, yet the effect is translated into a global optimization.
2. The essence of this mechanism is captured in six well-defined axioms that exhibit properties of global optimality, truthfulness, and utilitarianism.
3. The distinctive feature of these axioms is that when amassed together, their individual properties constrict into one system-wide performance enhancement property.
4. We use these axioms to obtain an efficient algorithm for the data replication problem.

The remainder of this paper is organized as follows. In Section 2 we present a formal description of the data replication problem. Section 3 focuses on describing the generalized axiomatic game theoretical mechanism along with its properties. Section 4 concentrates on modeling the mechanism for the data replication problem. The experimental results, related work, and concluding remarks are provided in Sections 5, 6, and 7, respectively.

2. The Data Replication Problem

Consider a distributed system comprising M servers, with each server 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 server i where $1 \leq i \leq M$. The M servers of the system are connected by a communication network. A link between two servers 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 servers S_i and S_j. If the two servers 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 server S_i to the server S_j. Without the loss of generality we assume that $c(i,j) = c(j,i)$. 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 server 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 server of the k-th object. Each primary server P_k, contains information about the whole replication scheme R_k of O_k. This can be done by maintaining a list of the servers where the k-th object is replicated at, called from now on the *replicators* of O_k. Moreover, every server S_i stores a two-field record for each object. The first field is its primary server P_k and the second the nearest neighborhood server NN_k^i of server S_i which holds a replica of object k. In other words, NN_k^i is the server 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 server of O_k. Another possibility is that $NN_k^i = P_k$, if the primary server is the closest one holding a replica of O_k. When a server 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 server can update every object. Updates of an object O_k are performed by sending the updated version to its primary server P_k, which afterwards broadcasts it to every server in 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 server NN_k^i. This cost is given by:

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

where $NN_k^i = \{Server\ j \mid j \in R_k \wedge \min c(i,j)\}$. 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 server P_k. This cost is given by the following equation:

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

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:

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

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:

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

Servers 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 server of O_k . Servers 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* $C_{overall}$, s*ubject to the storage capacity constraint*: $\sum_{k=1}^{N} X_{ik}o_k \le s_i \;\forall(1\le i\le M)$, *and subject to the primary copies policy:* $X_{P_k k}=1 \;\forall(1\le k\le N)$."

3. Axiomatic Game Theoretical Mechanism

In this section we use various building blocks to construct a generalized axiomatic game theoretical mechanism. We begin by defining a mechanism [3], which has two components: a) the *algorithmic output*, and b) the agents' *valuation functions*.

Definition 1 ([3])**:** *A mechanism is in which:*

1. *The system consists of M agents. Each agent i has some private data* $t^i \in \Re$ *. This is termed as the agent's true data or true value. Everything else in the mechanism is public knowledge.*
2. *The algorithmic output maps to each true data vector* $t = t^1 \ldots t^M$ *a set of allowed outputs* $x \in X$*, where* $x = x^1 \ldots x^M$.
3. *Each agent i's preferences are given by a real valued function* $v^i(t^i,x)$ *called valuation.*

Remarks: The valuation of an agent represents the quantification of its value from the output x, when its true data is t^i in terms of some predefined currency. For example, if the output of the mechanism is x and it hands the agent p^i amount of payment, then its utility becomes: $u^i = p^i + v^i(t^i,x)$.

We now focus on identifying a mechanism that allows the algorithmic output to optimize a given objective function. Below we give a refined definition of an optimization (minimization in our case) oriented mechanism.

Definition 2 ([3])**:** *An optimization oriented mechanism is one where:*

1. *The algorithmic output is given by a positive real valued objective function* $g(t,x)$*, and*
2. *a set of feasible outputs X.*

Thus, we require an output $x \in X$ that minimizes g, such that for any other output $x' \in X$, $g(t,x) \le g(t,x')$. This is fine as long as we can find a mechanism that can solve a given problem by assuring that the required algorithmic output occurs, when all the agents choose strategies that maximize their utility functions (a min-max procedure). Let a^{-i} denote a vector of strategies, not including agent i, *i.e*, $a^{-i} = (a^1,\ldots,a^{i-1},a^{i+1},\ldots,a^M)$. We can define a mechanism that is able to solve a utilitarian based minimization problem as:

Definition 3 ([24])**:** *A mechanism* ($m = (x(\cdot),p(\cdot))$) *is composed of* a) *an algorithmic output function* $x(\cdot)$*, and* b) *the payment function* $p(\cdot)$*. The mechanism should have the following properties:*

1. *The mechanism allows for each agent i a set of strategies* A^i*. It is up to the agent what strategy* ($a^i \in A^i$) *to adopt in order to have its utility function optimized.*
2. *The mechanism should provide an algorithmic output x derived from the output function, i.e.,* $x = x(a^i \ldots a^M)$.
3. *In order to motivate the agents, the mechanism should provide a payment* $p^i = p^i(a^1 \ldots a^M)$ *to each of the M agents.*

Remarks: Recall that $x = x^1 \ldots x^M$. It is then not difficult to see that agent i's algorithmic output can easily be obtained once the mechanism identifies the output x. It is possible that agents due to their selfish nature may alter the output of the algorithm in order to fervently gather more resources. Hence, we are required to enforce the mechanism to handle the special case of selfish agents by adding the following property to Definition 3.

Property 4: *The mechanism should be a representation of dominant strategies, and this is possible if for each agent i and each* t^i *there exists a dominant strategy* ($a^i \in A^i$)*, such that for all possible strategies of the other agents* a^{-i}*,* a^i *maximizes agent i's utility.*

Remarks: Literature survey reveals that the simplest of all the mechanisms that exhibits dominant strategies is the one where the agents' strategies are to report their true data. Such types of mechanisms are called *truthful* mechanisms, and they are based on the revelation principle [11]. This principle reports that for a given optimization problem, if there exists a truthful mechanism then pareto-optimality is guaranteed (by default). To align ourselves with Property 4 of Definition 3, we show that truth-telling is indeed a dominating strategy.

Lemma 1: *For agents to report their type (truth-telling) is a dominating strategy.*

Proof: Let (a^i,a^{-i}) denote a vector of strategies of all the M agents, *i.e.*, $(a^i,a^{-i}) = (a^1 \ldots a^M)$. Truth-telling would

mean that $a^i = t^i$. Then, for every $a^{i'} \in A^i$, if we define $x = x(a^i,a^{-i})$, $x' = x(a^{i'},a^{-i})$, $p^i = p^i(a^i,a^{-i})$, and $p^{i'} = p^{i'}(a^{i'},a^{-i})$, then $p^i+v^i(t^i,x) \geq p^{i'}+v^i(t^i,x')$, *i.e.*, $u^i \geq u^{i'}$. ■

We can use this result to couple it with a useful theorem reported in [23]. This will characterize the mechanism as truthful. Below we state the theorem.

Theorem 1 ([23])**:** *A mechanism that implements a given problem with dominant strategies is truthful.* ■

In Definition 3 we stated (Property 1) that a mechanism $m = (x(\cdot),p(\cdot))$ allows the agents to maximize their utilities. Since utilities enable the agents to express their preferences, it is important to identify an oracle that does exactly that. Literature survey revealed the following result, which we append as property 5 to the property list of Definition 3.

Theorem 2 ([24])**:** *A mechanism is called utilitarian if its objective function satisfies* $g(x,t) = \sum_i v^i(t^i,x)$. ■

Property 5: *The mechanism should have an objective function that satisfies* $g(x,t) = \sum_i v^i(t^i,x)$.

The above property is so useful that it can help us identify the two important components of a game theoretical mechanism. We state:

Theorem 3 ([10])**:** *A truthful mechanism m = (x(t),p(t)) belongs to the class of minimization utilitarian mechanisms if and only if*:

1. $x(t) \in \text{argmin}_x(\sum_i v^i(t^i,x))$.
2. $p^i(t) = \sum_{j \neq i} v^j(t^j,x(t)) + h^i(t^{-i})$, where $h^i(\cdot)$ is an arbitrary function of t^{-i}. ■

Remarks: Note that conditions 1 and 2 of Theorem 3 are the exact mathematical derivations of Properties 2 and 3 of Definition 3, respectively. Moreover, the mechanism stated in Theorem 3 takes in as argument t for both the algorithmic output and the payment function, *i.e.*, $m = (x(\cdot),p(\cdot))$ is now written as $m = (x(t),p(t))$. This is because of the mergence of Theorems 1 and 2. For convenience we shall now state the truthful mechanism in the axiomatic form (Figure 1).

We aim to use the above (discussed) axiomatic game theoretical mechanism to find solutions for the data replication problem (Section 3). The six axioms defined above will act as a cast for the data replication problem. In essence, we want a replica allocation mechanism that solves the data replication problem with the properties guaranteed by the six axioms.

4. Axiomatic Game Theoretical Replica Allocation Mechanism (AGT-RAM)

The directions laid down in Section 4 will be used to apply the axiomatic game theoretical mechanism to the data replication problem.

Ingredients (Axiom 1): The distributed system describe in Section 3 is considered and it consists of M agents. That is, each server is represented by an agent.

Axiomatic Game Theoretical Mechanism

Axiom 1 (*Ingredients*): A mechanism should have a) an algorithmic output specification, and b) agents' utility functions.

Axiom 2 (*Agent disposition*): Every agent has a private value termed as true data, everything else is public knowledge. This value along with a valuation function should reveal the preferences of the agent.

Axiom 3 (*Truthful*): The mechanism should have agents that project their dominant strategies.

Axiom 4 (*Utilitarian*): The mechanism's objective function should be to sum the agents' valuations.

Axiom 5 (*Motivation*): The mechanism should reward the agents with a payment. These payments are made in accordance to a specified function based on the algorithmic output.

Axiom 6 (*Algorithmic output*): The mechanism's algorithmic output should be a function that aids the agents to execute their preferences.

Figure 1: Axiomatic Game Theoretical Mechanism.

Assume for the time being that the feasible output (of the algorithm exists, and) are all partitions $x = x^1...x^M$ of the objects to the agents, where x^i is the set of objects allocated to agent i. Also assume that each agent i's utility function u^i exists. (In the subsequent text it will be clear what exactly x and u^i are.)

Agent disposition (Axiom 2): An agent holds two key elements of data a) the available server capacity b^i, and b) the cost of replication or valuation (CoR_k^i) of object O_k to the agent's server i. There are three possible cases:

1. DRP [π]: Each agent i holds the cost to replicate $CoR_k^i = t^i$ associated with each object O_k, where as the available server capacity and everything else is public knowledge.
2. DRP [σ]: Each agent i holds the available server capacity $b^i = t^i$, where as CoR_k^i and everything else is public knowledge.
3. DRP [π,σ]: Each agent i holds both the cost of replication and the server capacity $\{CoR_k^i,b^i\} = t^i$, where as everything else is public knowledge.

Remarks: Intuitively, if agents know the available server capacities of other agents, that gives them no advantage whatsoever. However, if they come about to know their CoR_k^i then they can modify their valuations and alter the algorithmic output. Note that an agent can only calculate CoR_k^i via the frequency of read and writes. Everything else such as the network topology, latency on communication lines, and even the server capacities can be public knowledge. Therefore, DRP[π] is the only natural choice.

Description of valuation (CoR_k^i): We can write CoR_k^i as follows:

$$CoR_k^i = \sum_{k\in\left(X_k^i=1\right)}\left[w_i^k o_k\left(\sum_{\forall j\in R_k, i\neq j} c\left(P_k, j\right)\right)\right], \quad (5)$$
$$+\sum_{k\in\left(X_k^i=0\right)}\left[r_i^k o_k c\left(i, NN_i^k\right) + w_i^k o_k c\left(i, P_k\right)\right]$$

which implies that if an agent replicates O_k (denoted in Equation 5 as $k \in X_i^k=1$), then the cost incurred due to reads is $0 = r_i^k o_k c(i,NN_i^k)$ since $NN_i^k = i$. The cost incurred due to local writes (or updates) is equal to zero since the copy resides locally, but whenever O_k is updated anywhere in the network, agent i has to continuously update O_k's contents locally as well. Therefore, the aggregate cost of writes is equivalent to $w_i^k o_k \Sigma_{\forall (j\in Rk),\ i\neq j} c(P_k,j)$. On the other hand if an agent does not replicate O_k (denoted in Equation 5 as $k \in X_i^k=0$), then the cost incurred due to reads is equal to $r_i^k o_k c(i,NN_i^k)$, and the cost incurred due to writes is equal to $w_i^k o_k c(i,P_k)$ since it only has to send the update to the primary server which then broadcasts the update based on R_k to the agents who have replicated the object.

Remarks: Equation 5 (above) captures the dilemma faced by an agent i when considering replicating O_k. If i replicates O_k then it brings down the read cost to zero, but now it has to keep the contents of O_k up to date. If i does not replicate O_k, then it reduces the overhead of keeping the contents up to date, but now it has to redirect the read requests to the nearest neighborhood server which holds a copy of O_k.

Truthful (Axiom 3): From Lemma 1, we know that truth-telling is a dominate strategy. From Axiom 2 (above) we know that $t^i = CoR_k^i$.

Utilitarian (Axiom 4): We proceed in two steps.

1. Let $y = \{x^i,o\}$ and $y' = \{x^i,o'\}$. In the context of the data replication problem, the valuation of an agent is give as $v^i(x,t^i) = \sum_{k\in xi} CoR_k^i$. This means that when an agent i is asked to express its preference over two objects o and o', it can do so by calculating the object impact factor, *i.e.*, the agent expresses its preference as min $\{\sum_{k\in y} CoR_k^i , \sum_{k\in y'} CoR_k^i\}$.
2. A utilitarian mechanism is one that has an objective function that is the sum of all agents' valuations, *i.e.*, $g(x,t) = \sum_i v^i(x,t^i) = \sum_i \sum_{k\in xi} CoR_k^i$.

We can see that Axiom 4 (Step 2) represents the data replication problem in its exact form as described in Equation 4. For a minute, let us ponder over both the representations. Equation 4 expresses the fact that in the data replication problem we have to find object allocations such that the object transfer cost (OTC) is minimized. Step 2 (in conjunction with Step 1) of Axiom 4 does exactly the same, *i.e.*, find the object allocations x^i such that the total cost of replication CoR_k^i is minimized

Motivation (Axiom 5): The motivational payment for each agent i is defined as $p^i(t) = \sum_{k\in xi(t)} \min_{i'\neq i} t^{i'} = \sum_{k\in xi(t)} \min_{i'\neq i} CoR_k^{i'}$, *i.e.*, for each object allocated to it, the agent is given payment equal to the overall second best cost of replication of any object to any server (a very strong incentive). The payment procedure also answers one of the pending questions of Axiom 1 (utility function), *i.e.*, $u^i = \sum_{k\in xi(t)} \min_{i'\neq i} t^{i'} + v^i(t^i,x)$.

Remarks: This motivational payment is need by the agents to cover the cost of hosting the object onto their server. This payment also ensures that the agents do indeed report true data. We justify this payment by analyzing the following cases:

1. *Over projection:* Agents in anticipation of more revenue over project their true data, but this does not help, as the agent who is allocated the object gets the second best payment.
2. *Under projection:* If every agent under projects their true data, that does not help either as the revenue would drop in proportion to the under projection.
3. *Random projection:* In this case the deserving agent would be at loss. Therefore, it is unlikely that a selfish agent would agree to project random true data.

For more details on the optimality of such type of payment procedure see [27]. In that paper, the authors have identified many such scenarios, but all fail to exploit this payment option.

Algorithmic output (Axiom 6): We now define an algorithm that actually aids the agents to execute their preferences. In the context of the data replication problem, the mechanism after gathering the true data from every agent, decides which object to be allocated to which agent. The mechanism is described in Figure 2. This also answers the pending question of Axiom 1 (algorithmic output function). Before we describe the algorithm, the data replication problem in the axiomatic game theoretical mechanism form is given as follows:

Find all the feasible outputs of the mechanism which are all the partitions $x=\{x^1 \ldots x^M\}$, where x is the entire replica allocation of the distributed system, and x^i is the set of replicas allocated to server i.

1. *The objective function of the mechanism is $g(x,t) = \sum_i v^i(x,t^i)$.*
2. *Agent i's valuation is $v^i(x,t^i) = \sum_{k\in xi} t^i$.*
3. *Agent i's true data is $t^i = CoR_k^i$.*
4. *Agent i's payment is $p^i(t) = \sum_{k\in xi(t)} \min_{i'\neq i} t^{i'}$.*
5. *Agent i's utility function is $u^i = p^i + v^i(t^i,x)$.*

Remarks: In the above problem formulation we did not mention that: 1) the agent i's valuation is actually to obtain the object impact (minimum cost of replication), 2) the agent i's server capacity constraint, and 3) the primary object constraints (both 2 and 3 are captured by x^i), but it is to be understood that they are indirectly embedded into the problem formulation.

Axiomatic Game Theoretical Replica Allocation Mechanism (AGT-RAM)

Initialize:
LS, L^i, T_k^i, *Mech*, *MT*

```
01 WHILE LS ≠ NULL DO
02   OMAX = NULL; MT = NULL; P^i = NULL;
03       PARFOR each S^i ∈ LS DO
04             FOR each O_k ∈ L^i DO
05                   T_k^i = compute (t_k^i);  /*compute the valuation corresponding to the desired object*/
06             ENDFOR
07          t_k^i = argmax_k(T_k^i);
08          SEND t_k^i to Mech; RECEIVE at Mech t_k^i in MT;
09       ENDPARFOR
10  OMAX = argmax_k(MT);    /*Choose the global dominate valuation*/
11  DELETE k from MT;
12  P^i = argmax_k(MT);          /*Calculate the payment*/
13  BROADCAST OMAX;
14  SEND P^i to S^i;       /*Send payments to the agent who is allocate the object OMAX*/
15  Replicate O_OMAX;
16  b^i=b^i - o_k;              /*Update capacity*/
17  L^i = L^i - O_k;            /*Update the list*/
18  IF L^i = NULL THEN SEND info to Mech to update LS = LS - S^i;      /*Update mechanism players*/
19       PARFOR each S^i ∈ LS DO
20            Update NN^i_OMAX          /*Update the nearest neighbor list*/
21       ENDPARFOR              /*Get ready for the next round*/
22 ENDWHILE
```

Figure 2: Pseudo-Code for Axiomatic Game Theoretical Replica Allocation Mechanism (AGT-RAM).

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 server 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 b^i. Moreover, if server 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 servers 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 08) to the mechanism. The mechanism receives all the corresponding entries, and then chooses the best 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 payments made to the agent. The mechanism progresses forward till there are no more agents interested in acquiring any data for replication.

The above discussion allows us to deduce the following two results about the mechanism.

Theorem 4: *In the worst case AGT-RAM takes* $O(MN^2)$ *time.*

Proof: The worst case scenario is when each server has sufficient capacity to store all objects. In that case, the while loop (Line 01) performs MN iterations. The time complexity for each iteration is governed by the two **PARFOR** loops (Lines 03 and 19). The first loop uses at most N iterations, while the send loop performs the update in constant time. Hence, we conclude that the worst case running time of the mechanism is $O(MN^2)$. ■

Theorem 5: *AGT-RAM is a truthful mechanism.*

Proof: The algorithmic output is an allocation that minimizes the utilitarian function $\sum_i v^i(x,t^i)$. Let h^{-I} be $\sum_k \min_{i' \neq i} t^{i'}$, then $\sum_{i' \neq i} v^{i'}(x,t^{i'}) + h^{-i}$ is exactly the mechanism's payment function. It is also evident that truth-telling is the only dominate strategy. For simplicity let us consider the case for only one object. The argument for $k>1$ is similar. Let d denote the declarations and t their real types. Consider the case where $d^i \neq t^i$ (i=1,2). If $d^i > t^i$, then for d^{3-i} such that $d^i > d^{3-i} > t^i$, the utility for agent i is $t^i - d^i < 0$, which should have been zero in the case of truth-telling. ■

5. Experiments, 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. AGT-RAM was implemented using Ada and Ada GNAT's distributed systems annex GLADE [25].

To establish diversity in our experimental setups, the network connectively was changed considerably. We used GT-ITM [2] for the network topologies, the procedure for which is as follows: A random graph $G(M,P(\text{edge} = p))$ with $0 \leq p \leq 1$ contains all graphs with nodes (servers) M in which the edges are chosen independently and with a probability p. The pure random topologies were obtained with p = {0.4, 0.5, 0.6, 0.7, 0.8}. In each of these topologies the distance between two serves was reversed mapped to the communication cost of transmitting a 1kB of data and the latency on a link was assumed to be 2.8×10^{-8} m/s (copper wire).

To evaluate the replica allocation methods under realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 web server [2]. Each experimental setup was evaluated thirteen times, *i.e.*, only the Friday (24 hours) logs from May 1, 1998 to July 24, 1998. (The Friday logs have the heaviest traffic compared to any other day of the week.) To process the logs, we wrote a script that returned: only those objects which were present in all the logs (25,000 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 server was mimicked by choosing random locations. The capacities of the servers 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 miscellanies to benchmark object updates. The updates were randomly pushed onto different servers, 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.

Since the access logs are of the year 1998, we first use Inet [5] topology generator to estimate the number of nodes in the network. This number came up to be 3718, *i.e.*, there were 3718 AS-level nodes in the Internet at the time when the Soccer World Cup 1998 was being played. Therefore, we set the upper bound on the number of servers in the system at M = 3718.

Comparative algorithms: 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. For fine-grained replication, the algorithms proposed in [15], [16], [19], [21], and [26] are the only ones that address the problem domain similar to ours. We select from [26] the greedy approach (Greedy) for comparison because it is shown to be the best compared with 4 other approaches (including the proposed technique in [19]); thus, we indirectly compare with 4 additional approaches as well. Algorithms reported in [16] (the efficient branch and bound based technique Aε-Star), [15] (Dutch (DA) and English auctions (EA)) and [21] (Genetic based algorithm (GRA)) are also among the chosen techniques for comparisons. Unfortunately, space limitations do no permit us to provide the detailed workings of these algorithms; however, we encourage the readers to obtain an insight on the comparative techniques from the referenced papers.

Performance metric: The solution quality was measured in terms of network communication cost (OTC percentage) that was saved under the replica scheme found by the replica allocation methods, compared to the initial one, *i.e.*, when only primary copies exists.

Comparative analysis: We observe the effects of increase in storage capacity. 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 Figure 3, which shows the performance of the algorithms. GRA once again performed the worst. The gap between all other approaches was reduced to within 15% of each other. AGT-RAM and Greedy 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 (49%) followed by Greedy with 44%. 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 18%, resulted in 4 times (on average) more replicas for all the algorithms.

Next, we observe the effects of increase in the read and write frequencies. Since these two parameters are complementary to each other, we describe them together. To observe the system utilization with varying read/write frequencies, we kept the number of servers and objects 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

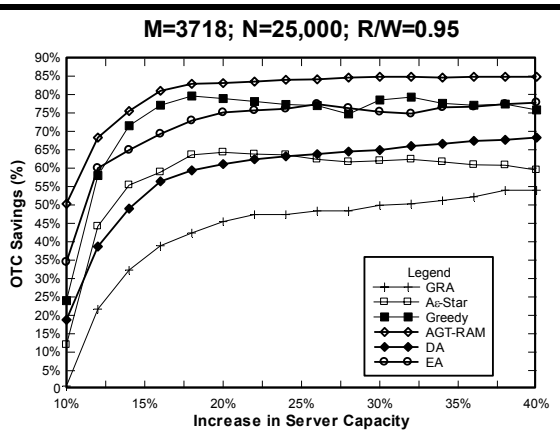

Figure 3: OTC savings versus server capacity.

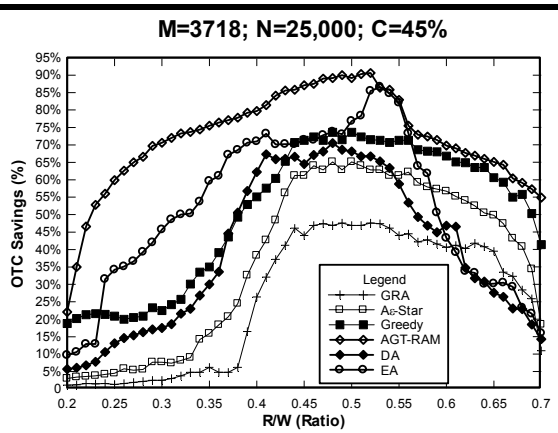

Figure 4: OTC savings versus read/write ratio.

Table 1: Running time of the replica placement methods in seconds [*C*=45%, *R/W*=0.85].

Problem Size	Greedy	GRA	Aε-Star	AGT-RAM	DA	EA	Improvement brought by AGT-RAM (%)	
M=2500, *N*=15,000	310.14	491.00	399.63	**186.12**	345.14	356.44	**39.98**	[=((310.14-186.12)/310.14)×100]
M=2500, *N*=20,000	330.75	563.25	442.66	**202.85**	354.94	368.94	38.67	[=((330.75-202.85)/330.75)×100]
M=2500, *N*=25,000	357.74	570.02	465.52	**242.23**	368.43	394.57	32.29	[=((357.74-242.23)/357.74)×100]
M=3000, *N*=15,000	452.22	671.68	494.60	**285.43**	475.60	489.76	36.88	[=((452.22-285.43)/452.22)×100]
M=3000, *N*=20,000	467.65	726.75	498.66	**286.75**	492.69	531.45	38.68	[=((467.65-286.75)/467.65)×100]
M=3000, *N*=25,000	469.86	791.26	537.56	**305.64**	501.51	543.08	34.95	[=((469.86-305.64)/469.86)×100]
M=3718, *N*=15,000	613.27	883.71	753.87	**370.43**	668.25	678.61	39.59	[=((613.27-370.43)/613.27)×100]
M=3718, *N*=20,000	630.39	904.20	774.31	**398.87**	694.68	702.15	36.72	[=((630.39-398.87)/630.39)×100]
M=3718, *N*=25,000	646.98	932.38	882.43	**405.76**	715.02	745.92	37.28	[=((646.98-405.76)/646.98)×100]

Table 2: Average OTC (%) savings under some randomly chosen problem instances.

Problem Size	Greedy	GRA	Aε-Star	AGT-RAM	DA	EA	Improvement brought by AGT-RAM (%)	
M=100, *N*=1000 [*C*=20%,*R/W*=0.75]	71.46	85.77	86.28	**88.12**	73.14	77.56	2.08	[=((88.12-86.28)/88.12)×100]
M=200, *N*=2000 [*C*=20%, *R/W*=0.80]	84.29	78.30	79.02	**84.95**	73.56	76.15	0.78	[=((84.95-84.29)/84.95)×100]
M=500, *N*=3000 [*C*=25%, *R/W*=0.95]	68.50	70.97	67.53	**72.15**	70.15	68.14	1.64	[=((72.15-70.97)/72.15)×100]
M=1000, *N*=5000 [*C*=35%, *R/W*=0.95]	88.09	67.56	78.24	**88.21**	70.86	66.80	0.14	[=((88.21-88.09)/88.21)×100]
M=1500, *N*=10,000 [*C*=25%, *R/W*=0.75]	89.34	52.93	76.11	**90.25**	62.48	74.13	1.01	[=((90.25-89.34)/90.25)×100]
M=2000, *N*=15,000 [*C*=30%, *R/W*=0.65]	67.93	51.02	52.42	**73.25**	66.19	63.84	7.26	[=((73.25-67.93)/73.25)×100]
M=2500, *N*=15,000 [*C*=25%, *R/W*=0.85]	77.35	71.75	73.59	**83.21**	70.36	72.01	7.04	[=((83.21-77.35)/83.21)×100]
M=3000, *N*=20,000 [*C*=25%, *R/W*=0.65]	76.22	65.89	73.04	**83.01**	72.16	70.53	**8.18**	[=((83.01-76.22)/83.01)×100]
M=3500, *N*=25,000 [*C*=35%, *R/W*=0.50]	66.04	59.04	67.01	**72.15**	62.20	63.57	7.12	[=((72.15-67.01)/72.15)×100]
M=3718, *N*=25,000 [*C*=10%, *R/W*=0.40]	76.34	63.19	76.02	**77.12**	75.91	76.10	1.01	[=((77.12-76.34)/77.12)×100]

the replicas be placed as close as to the primary server 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 plot in Figure 4 shows the results of read/write ratio against the OTC savings. A clear classification can be made between the algorithms. AGT-RAM and Greedy incorporate the increase in the number of reads by replicating more objects and thus savings increased up to 88%, while GRA gained the least of the OTC savings of up to 42%. To understand why there is such a gap in the performance between the algorithms, we should recall that GRA specifically depends on the initial selection of gene population (for details see [21]). 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, AGT-RAM, DA, EA, Aε-Star and Greedy never tend to deviate from

their global (or social) view of the problem.

Lastly, we compare the termination time of the algorithms. Various problem instances were recorded with C=45% and R/W=0.85. The entries in Table 1 made bold represent the fastest time recorded over the problem instance. It is observable that AGT-RAM terminated faster than all the other techniques, followed by Greedy, DA, EA, Aε-Star, and GRA.

Table 2 shows the quality of the solution in terms of OTC percentage for 10 problem instances (randomly chosen), each being a combination of various numbers of server and objects, with varying storage capacity and update ratio. For each row, the best result is indicated in bold. The proposed AGT-RAM steals the show in the context of solution quality, but Greedy and Aε-Star do indeed give a good competition.

In summary, based on the solution quality alone, the replica allocation methods can be classified into four categories: 1) High performance: AGT-RAM; 2) Medium-High performance: Greedy; 3) Medium performance: Aε-Star and DA; 5) Low performance: EA and GRA. Considering the execution time, AGT-RAM and Greedy did extremely well, followed by DA, EA, Aε-Star and GRA.

6. Related Work

The data replication problem is an extension of the classical file allocation problem (FAP). Chu [6] studied the file allocation problem with respect to multiple files in a multiprocessor system. Casey [3] extended this work by distinguishing between updates and read file requests. Eswaran [8] proved that Casey's formulation was NP-complete. In [21] Mahmoud *et al.* provide an iterative approach that achieves good solution quality when solving the FAP for infinite server capacities.

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 major works which directly or indirectly deal with the data replication problem using game theoretical techniques. The first work [8] is mainly on caching and uses an empirical model to derive Nash equilibrium. The second work [17] focuses on mechanism design issues and derives an incentive compatible auction for replicating data on the Web. The third work [18] 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 non-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 self-interested manner, and 3) deriving pure Nash equilibrium and pure strategies for the agents.

Readers are encouraged to see [20] for a comprehensive survey on the conventional replica placement techniques.

7. Conclusions

This paper proposed a semi-distributed axiomatic game theoretical replica allocation mechanism (AGT-RAM) for object based data replication in large distributed computing systems such as the Internet. AGT-RAM is a protocol for automatic replication and migration of objects in response to demand changes. It aims to place objects in the proximity of a majority of requests while ensuring that no hosts become overloaded.

The infrastructure of AGT-RAM was designed such that, each server was required to present a list of data objects that if replicated onto that server would bring the communication cost to its minimum. These lists were reviewed at the central decision body which gave the final decision as to what object are to be replicated onto what servers. This semi-distributed infrastructure takes away all the heavy processing from the central decision making body and gives it to the individual servers. For each object, the central body is only required to make a binary decision: (0) not to replicate or (1) to replicate.

To compliment our theoretical results, we compared AGT-RAM with five conventional replica allocation methods namely: (1) branch and bound, (2) greedy, (3) genetic, (4) English, and (5) Dutch auctions. The experimental setups were designed in such a fashion that they resembled real world scenarios. We employed GT-ITM and Inet to gather various Internet topologies and used the traffic logs collected at the Soccer World Cup 1998 website for mimicking user access requests. The experimental study revealed that the proposed AGT-RAM technique improved the performance relative to other conventional methods in four ways. First, the number of replicas in a system was controlled to reflect the ratio of read versus write access. To maintain concurrency control, when an object is updated, all of its replicas need to be updated simultaneously. If the write access rate is high, there should be few replicas to reduce the update overhead. If the read access rate is overwhelming, there should be a high number of replicas to satisfy local accesses. Second, performance was improved by replicating objects to the servers based on locality of reference. This increases the probability that requests can be satisfied either locally or within a desirable amount of time from a neighboring server. Third, replica allocations were made in a fast algorithmic turn-around time. Fourth, the complexity of the data replication problem was decreased by multifold. AGT-RAM limits the complexity by partitioning the complex global problem of replica allocation, into a set of simple independent sub problems. This approach is well suited to the large distributed

computing systems that are composed of autonomous agents which do not necessarily cooperate to improve the system wide goals. All the above improvements were achieved by a simple, semi-distributed, and autonomous AGT-RAM.

As future work, we would extend the semi-distributed model to regional autonomous, self-governed and self-repairing mechanisms. That is, the current system model would be broadened to incorporate regional or hierarchical mechanisms. This would enable the system to be less vulnerable to the failures of a single mechanism, and in turn would open the realms of devising hierarchical games, where in each level either a cooperative or non-cooperative game could be played to replicate data objects.

References

[1] I. Ahmad and A. Ghafoor, "Semi-Distributed Load Balancing for Massively Parallel Multicomputer Systems," *IEEE Trans. Software Engineering*, 17(10), 987-1004, 1991.

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

[3] D. Campbell, *Resource Allocation Mechanisms*, Cambridge University Press, 1987.

[4] R. Casey, "Allocation of Copies of a File in an Information Network," *in Proc. Spring Joint Computer Conf.*, IFIPS, 1972, pp. 617-625.

[5] H. Chang, R. Govindan, S. Jamin and S. Shenker, "Towards Capturing Representative AS-Level Internet Topologies," *Computer Networks Journal*, 44(6), pp 737-755, 2004.

[6] K. Chandy and J. Hewes, "File Allocation in Distributed Systems," in *Proc. of the International Symposium on Computer Performance Modeling, Measurement and Evaluation*, 1976, pp. 10-13.

[7] W. Chu, "Optimal File Allocation in a Multiple Computer System," *in IEEE Trans. on Computers*, C-18(10), pp. 885-889, 1969.

[8] 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 Symposium on Principles of Distributed Computing*, 2004, pp. 21-30.

[9] K. Eswaran, "Placement of Records in a File and File Allocation in a Computer Network," *Information Processing Letters*, pp. 304-307, 1974.

[10] J. Green and J. Laffont, "Characterization of Satisfactory Mechanisms for the Revelation of Preferences for Public Goods," *Econometrica*, 45(2), pp. 427-438, 1977.

[11] T. Groves, "Incentives in Teams," *Econometrica*, vol. 41, pp. 617-631, 1973.

[12] S. Hakimi, "Optimum Location of Switching Centers and the Absolute Centers and Medians of a Graph," *Operations Research*, vol. 12, pp. 450–459, 1964.

[13] 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.

[14] 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.

[15] S. Khan and I. Ahmad, "Internet Content Replication: A Solution from Game Theory," Department of Computer Science and Engineering, University of Texas at Arlington, Technical Report, CSE-2004-5, 2004.

[16] S. Khan and I. Ahmad, "Heuristic-based Replication Schemas for Fast Information Retrieval over the Internet," in *Proc. of 17th International Conference on Parallel and Distributed Computing Systems*, 2004, pp. 278-283.

[17] 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.

[18] N. Laoutaris, O. Telelis, V. Zissimopoulos and I. Stavrakakis, "Local Utility Aware Content Replication," to appear in *IFIP Networking Conference*, 2005.

[19] 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.

[20] T. Loukopoulos, D. Papadias and I. Ahmad, "An Overview of Data Replication on the Internet," in *Proc. of International Symposium on Parallel Architectures, Algorithms and Networks*, 2002, pp. 31-38.

[21] 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, 2004.

[22] S. Mahmoud and J. Riordon, "Optimal Allocation of Resources in Distributed Information Networks," *ACM Trans. on Database Systems*, 1(1), pp. 66-78, 1976.

[23] A. Mas-Collel, W. Whinston and J. Green, *Microeconomic Theory*, Oxford University Press, 1995.

[24] N. Nisan and A. Ronen, "Algorithmic Mechanism Design," in *Proc. of 31st ACM Symposium on Theory of Computation*, 1999, pp. 129-140.

[25] L. Pautet and S. Tardieu, "GLADE: A Framework for Building Large Object-Oriented Real-Time Distributed Systems," in *3rd International Symposium on Object-Oriented Real-Time Distributed Systems*, 2000, pp. 244-251.

[26] L. Qiu, V. Padmanabhan and G. Voelker, "On the Placement of Web Server Replicas," in *Proc. of the IEEE INFOCOM*, 2001, pp. 1587-1596.

[27] S. Saurabh and D. Parkes, "Hard-to-Manupilate VCG-Based Auctions," Available at: http://www.eecs.harvard.edu/ economics/pubs/hard_ to_manipulate.pdf

[28] L. Tesfatsion, "Agent-based Computational Economics: Growing Economies from the Bottom Up," *Artificial Life*, vol. 8, no. 1, pp. 55-82, 2002.