HEURISTICS-BASED REPLICATION SCHEMAS FOR FAST INFORMATION RETRIEVAL OVER THE INTERNET

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

Abstract

Internet today, has transformed into a global information hub. The increase in its usage and magnitude have sparkled various research problems. Because of the diverse user population, along with the frequency of access requests, the need for data replication in order; to decrease latency and communication cost, to optimize bandwidth, to effectively utilize the storage space, and to add reliability to the system has emerged to the surface along with object caching. In this paper we address the fine-grained replication of data among a set of Internet sites and develop its cost model. We solve this problem by proposing data placement algorithms. Four of our proposed techniques are based on the A-star state space searching algorithm. The optimal A-star based technique is complemented by three sub-optimal heuristics, and two natural (greedy based) selection algorithms: Local and Global Min-Min. These algorithms are effective in various environments providing vendors and users with the choice of algorithms that guarantee fast or optimal or both types of solutions.

1 INTRODUCTION

The Internet conceived as a mode of communication among a group of universities in the late 60's, has now evolved into the ultimate source of information. Thus, a particular server is experiencing increase in traffic on daily basis, and suffers from bandwidth saturation especially during the peak hours. Moreover, peak hours vary due to the time differences between different users and within one time zone due to the user behavior [1]. Replication is therefore a more viable solution than caching, since it provides consistency and reduces system overload considerably [4]. Replication was originally designed and used for manual mirroring web sites. Later, it took on the task of distributing the requests among a static set of mirrors [10]. Currently we are confined with manual selection of replicas, but with the rapid growth of the Internet and the every increasing need for such systems [1], dynamic replication would be a popular solution. The existing techniques have been observed to be expensive in terms of network communication and quality of solution [10]. This is due to the fact that point-to-point communication among nodes is assumed, when multi-cast communication is available [2]. In all such techniques authors have undertaken certain assumptions since the generalized data replication problem is NP-complete [9]. Thus every technique may be complete and/or optimal over the assumed problem domain.

All the initial work on the data replication over the internet assumed coarse-grained replication model, where the entire contents of the sites were replicated. A comprehensive survey on coarse-grained replication strategies can be found in [10].

In this paper, we formulate and focus on the fine-grained replication problem, *i.e.*, we allow reallocation of objects that are accessed and not the entire site. This approach has many advantages, such as; it saves the server memory capacity by only moving those object that are actually required to be moved, the problem definition is highly scalable and robust. The generalized fine grain replication is known to be NP-complete not only for general graphs, but also for partitioned graphs [8]. This particular line of research, is gaining much popularity. In [9], they analyzed both static such as a modified Greedy based approach, Evolutionary method based on Genetic algorithms, and Adaptive Genetic approach. Experimental results revealed that Genetic approach outperformed on every occasion than all the other approaches. The work was further extended with comparisons to Linear Programming, and Linear Integer Programming approach. Nevertheless, fine-grained replication shows more flexibility and scalability then course-grained replication. A brief introduction to the existing work can be found in [10]. A much earlier survey on replication and its applications to the Internet can be found in [11].

This paper addresses the problem of web content replication as a *fine-grain* (object based) *replication* [10] with the aim to find solutions (optimal and sub-optimal) in a fast turn-around time. We use the A-star state space searching technique to identify optimal solution(s). This approach is complemented by three sub-optimal A-star based heuristics, which sacrifice some solution quality but incur fast execution time and do not suffer from memory overflow problems associated with A-star type algorithms [7]. We also propose two natural selection algorithms: Local and Global Min-Min. We evaluate and compare the proposed algorithms by analyzing the system utilization. The main purpose of this study is to provide vendors/users with the choice of algorithms that guarantee fast or optimal or both types of solutions. The optimal solutions would be more suitable for static replication, while fast solutions are important for dynamic systems that require frequent updates.

The remainder of this paper is organized as follows. Section 2 describes the data replication problem and the system model. Section 3 describes the placement policies. The experimental results and concluding remarks are provided in Sections 4 and 5 respectively.

Table 1: Notations and their meanings.

Symbols	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.
SGRG	Self Generate Random Graphs
GT-ITM PR	Georgia Tech Internetwork Topology Models Pure Random
GT-ITM W	GT-ITM Waxman
SGFCGUD	Self Generated Fully Connected Graphs with Uniform Distribution
SGFCGRD	SGFCG with Random Distribution
SGRGLND	SGFCG with Lognormal Distribution

2 THE PROBLEM DESCRIPTION

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 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 very common assumption (e.g. see [9]) 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 during a certain time period. 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 Data Replication Problem (DRP) 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. First, is the RC created from 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:

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

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:

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

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

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

Let X_{ik}=1 if Si 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:

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

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 Data Replication Problem (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: $\sum_{k=1}^{N} X_{ik} o_k \leq s_i \forall (1 \leq i \leq M)$, and *subject to the primary copies policy*: $X_{P_k k} = 1 \forall (1 \leq k \leq N)$.

In the generalized case, DRP is essentially a constraint optimization problem, reducible to the *Knapsack* problem, and without the storage constraint, to the *minimum k-median* problem [9].

3 REPLICA PLACEMENT TECHNIQUES

3.1 A-star

The A-star based searching technique for the Data Replication Problem (DRPA-star) starts from an assignment *P*, and explores all the *potential* options of assigning an object to a site. With proper pruning techniques used against the constraint(s) *C*, only the assignments in the admissible head set are explored. If the new solution is consistence with the constraint, it is added to the Expansion Tree (ET), otherwise the solution is pruned. In order to avoid memory overflow, we limit the ET to 1000 active solution (state) space allocations. This is very common technique used to overcome the memory overflow problem associated with A-star type algorithms. For details see [7]. Moreover, the candidate objects assignments are ordered (in a linked list termed as the OPEN list), such that the smallest projected cost of allocation is expanded first. Thus, we can terminate our expansion when the solution for the data replication problem is obtained, or there are no more candidate allocations left in the ET. In either case optimality is always guaranteed. DRPA-star uses the following heuristic: 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 *t* be the global minimum of an object's replication cost. Thus, we define the minimum of such cost as a set: $T=min_{0\leq j\leq N-1}(t(O_k, S_i))$, $\forall O_k \in U$. For a node *n*, let *mmk*(*n*) define the maximum element of set *T* (the max-min replication cost). *mmk*(*n*) then represents the best possible replica allocation without the unrealistic assumption that every object in *U* can be replicated to a site in *M* without a conflict. Thus we give our heuristic for the replication cost as follows: $h_{cost}(n)=max(0,[mmk(n)-g(n)])$.

Lemma 1: *DRPA-star grows and requires sub-exponential time and space.*

Proof: Let *P* be the expanded paths (partial or complete solution) in the ET, then the space required by the DRPA-star is *P* and the time required by DRPA-star is $dP(h+\log(P))$. Where *d* is the degree of the network, *h* is the depth at which the solutions are identified, and the $\log(P)$ factor identifies the growth of the search tree. Now if error in the heuristic grows no faster than log of the optimal cost of the solution. A-star has been proven to be sub-exponential [7]. Since DRPA-star due to its pruning is far more efficient than A-star, DRPA-star will also grow sub-optimally. We give the relation of sub-optimality as: $OPT_{cost}-A_{cost}\leq O(\log(OPT_{cost}))$, where OPT_{cost} is the optimal cost of state space search expansion, and A_{cost} is the admissible cost. We can thus say that: $P\leq Mhd\leq dM^2$. For an average case analysis DRPA-star uses space equivalent to *Mhd*, and thus the running time would be $Mhd^2(h+\log(Mhd))$.

3.2 A-star based heuristics

We now present three heuristics (sub-optimal A-star) algorithms, refereed to hereafter as SA1, SA2, SA3. The name SA comes from Sub-optimal Assignments. The main purpose is to design algorithms that converge to solution faster and overcome the high memory requirements associated with A-star type algorithms [7]. The basic idea behind these algorithms is that when the search process reaches a certain depth in the search tree, some search path(s) can be avoided (some tree nodes can be discarded) without moving far from the optimal solution. In SA1, when the algorithm (DRPA-star) selects a node that belongs to level *R* or below, it generates only the best successors (lowest expansion cost) of it. All the other successors except the best one are discarded. In SA2, when the depth level *R* is reached for the very first time, all the successors except the minimum cost are discarded among all the nodes marked for expansion. In SA3, the discarding is done similar to SA2 except that now the nodes are removed from the ET. For instance, if *n* nodes are generated, then all of them are inserted in the ET, and the *n-1* high cost nodes are discarded. These techniques will not suffer from memory overflow, since at level *R*, for every node taken out of the ET for expansion, only one node is inserted. Also the running time is reduced by many folds since the algorithm expands/explores less number of nodes when it reaches *R*.

3.3. Local Min-Min

Let O_k and S_i represent the set of objects and sites in the system. Let *U* be the set of unassigned objects to a site S_i. Let U_{min} define the minimum replication cost of the objects to be assigned to a particular site. The assignment is made in the ascending order of set *U*. If there is a tie among two objects, then the tie is broken by the minimum object size, hence the name Min-Min. Since we do the assignment iteratively for every object and do not consider the effects of the choice of an object to a site with respect to other sites, we call it Local Min-Min (LMM).

Lemma 2: *LMM converges in* $O(MN(\log N))$ *and requires linear space.*

Proof: The most expensive part of the algorithm is the sorting. Since we iterate *N* objects over all the *M* sites repeatedly, the entire step would take $MN(\log N)$. The assignment part of the algorithm would at most take O(MN). Thus the most expensive part of the algorithm would have the bound of O(MN(log N)). It is not difficult to see that it would take linear space for completion. Intuitively, we only load the N objects and iteratively assign then to a site. Therefore at each step we need at most O(M+N) memory.

3.4 Global Min-Min

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 object is chosen on random. 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).

Lemma 3: *GMM requires* O(M^2N^2(log N)) *running time and* Ω(MN) *space.*
Proof: We can efficiently obtain the min(*sort*(T)) in O(MN(log N)). This would be the time required to assign the first object. Since there would be at most $M\times N$ unique entries in the *Replication Cost Matrix*, we are therefore required to exploit all the $M\times N$ candidates. Thus, the total time required would be O(M^2N^2(log N)). GMM would have to load the entire $M\times N$ matrix for sorting thus it would take no less than Ω(MN) of memory.

4 EXPERIMENTAL EVALUATIONS

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, is measured according to the RC percentage that is saved under the replication scheme found by the algorithms, compared to the initial one, *i.e.*, when only primary copies exist. In all the experiments for the proposed A-star based sub-optimal heuristics, the cutoff R was set at: $R=\lfloor d/2 \rfloor$, where d represents the depth of the tree (number of objects).
The network architecture is generated as follows. First, the number of sites M and objects N are defined. To establish diversity in our experimental setups, the network connectively is 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 is equivalent to the communication cost. Table 2 summarizes the various techniques used to gather forty-five various topologies for the 100 node networks. It is to be noted that the parameters vary for network with lesser/larger number of nodes. All the results reported, represent the average performance over all the topologies.

To evaluate our proposed replication techniques on realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 website [3]. 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 (13×5) 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. The primary replicas' original site was mimicked by choosing random locations. The capacities of the sites C% are 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 helps to instill enough diversity to benchmark object updates. The updates are randomly pushed onto different sites, and the total system update load is measured in terms of the percentage update requests U% compared that to the initial network with no updates.

Table 2: Parameters for topologies with 100 nodes.

Topology	Mathematical Representation	Parameter 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 [5] (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 [5] (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 [6].	μ={8.455,9.345,9.564}, σ={1.278,1.305,1.378}.

Table 3: Running time (sec.) of proposed techniques.

Problem Size	SA1	SA2	SA3	LMM	GMM
M= 500, N= 1350	560	389	526	429	432
M= 500, N= 1400	609	468	609	454	452
M= 500, N= 1450	662	583	662	471	502
M= 500, N= 1500	707	641	740	497	532
M= 500, N= 1550	804	698	807	503	583
M= 500, N= 2000	846	725	901	517	634

Table 3 shows the execution times of all except the DRPA-star technique. The number of sites was kept constant at 500, and the number of objects was varied from 1350 to 2000. For DRPA-star we choose a smaller setup as it could not handle the massive data processing.

DRPA-star terminated with a massive time of 156 minutes (9356 sec.) with 250 objects and 100 sites setup. With the maximum configuration, *i.e.*, 2000 objects and 500 sites; SA1, SA2 and SA3 performed well with termination times of 846 sec., 725 sec., and 901 sec. respectively. LMM and GMM on the other had outperformed every other placement policy with a turn around time of 517 sec. and 634 sec. respectively.

It is a fact that the more the replicas the more reliable is the system. However, this does not guarantee that the obtained solution is of a high quality, *i.e.*, minimum RC. Our first judgment on the proposed techniques would be to see how much they can replicate. It is to be noted that we do not relax our constraints of minimizing the total replication cost or the storage. The creations of replicas are dependent on two major factors, *i.e.*, the number of objects available and the number of sites in the network. Thus, we perform two sets of experiments. The update ratio is fixed for the entire test at 10%. For the first type of experiment, we vary the number of sites in the network, while the number of objects is kept constant at 2000. Figure 1 shows the amount of replicas created as compared to the initial setup. SA3 with 14.54 and GMM with 10.1 outperformed the rest of the heuristics. LMM, SA1 and SA2 performed poorly with at most 6.23 times replica creation compared to the initial network setup. Next we kept the number of sites constant at 500 and studied the effects of increasing number of objects in the system. The number of objects was varied from 1200 to 2000. Here we can observe (Figure 2) a clear difference between high and low performance algorithms. SA1 and SA2 performed the worst with a drop of 40% in its replica creation. Unexpectedly LMM retained its replicas. This poor performance of SA type algorithms is credited to the bound *R*. With the increase in the number of objects the available choices also increase, but the bound remains the same. It is to be noted that the bound is on the depth of the search tree and not the degree, but the degree is indirectly affected when the bound *R* is met, causing a low performance. GMM performed extremely well with a loss of only 5% in its replicas.

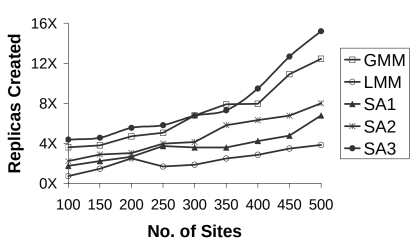

Figure 1: Replicas created vs. no. of sites.

Our formulation of the data replication problem also caters for disk storage constraints. It is understandable that with the increase of available space, the algorithms will try to accommodate as many objects as possible. Since the objects' size remain constant at all times, with the increase in the capacity of the sites the choice of allocations also increase. In Figure 3 every algorithm gained a slight saving with the increase of capacity from 0% to 100%. The major gain was observed in the case of LMM (19%).

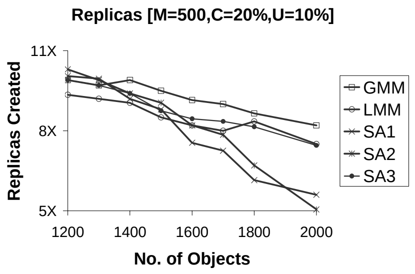

Figure 2: Replicas created vs. no. of objects.

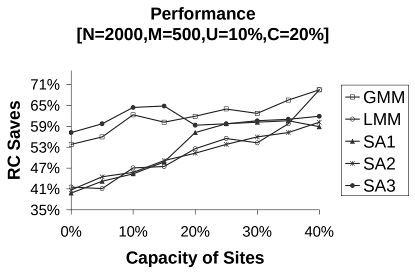

Figure 3: RC savings vs. site capacity.

The effects of reads on our computational model were studied next. The purpose of such a study is to see how flexible our proposed model is with the increasing number of user read requests. In this setup, we increase the read requests from 0% to 100% on all the objects*, i.e.*, it will be an aggregate read increase on the system. We keep N=2000, M=500, U=15%, and C=20%. Figure 4 shows the test runs. All most every algorithm showed a consistent performance. Increase in savings (approx. 8%) was observed in LMM, SA1, and SA2. SA3 and GMM on the other hand did not gain much, but showed a stable performance.

Similar to reads, we also studied the effects of updates (writes). To give a brief idea on how we perform the tests, we record the RC savings on a particular configuration and then we send update requests to P_i on random. P_i upon receiving the request will broadcast an update request to the sites containing the replicas. Thus, we can measure the difference in load on the system before and after the update request(s). This concept is analogous to measuring the communication cost in the system. From server point of view this is important since we want the RC savings to be at least maintained if not gained. The higher the difference the higher would be the communication cost which in turn means the system would be less stable. Figure 5 shows the test results. We can observe a clear difference in the solution quality between SA3, GMM and the rest of the algorithms. GMM is only off by approximately 10% of GMM. Interesting to see is the poor performance of both LMM and SA2 which with 40% increase in the updates show a mere 2% and 9% RC savings.

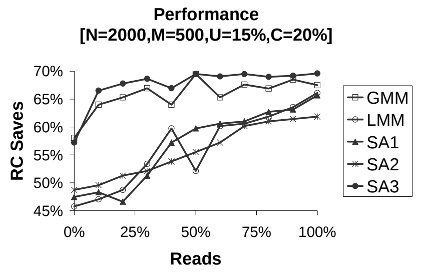

Figure 4: RC savings vs. reads.

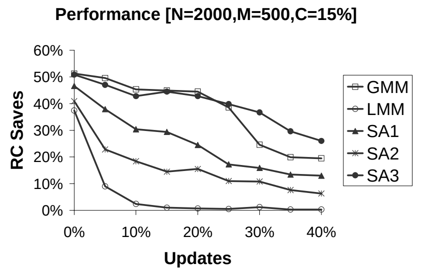

Figure 5: RC savings vs. updates.

Table 4: Summary of the results.

Algorithm	Replicas		RC Savings						Score	Rank
	Sites	Objects	Sites	Objects	Capacity	Object Size	Reads	Updates		
DRPA-star	1	1	1	1	1	1	1	1	08	1
LMM	5	4	6	6	5	6	6	6	44	6
GMM	3	2	3	4	3	4	2	4	25	3
SA1	6	6	5	5	6	5	5	5	43	5
SA2	4	5	4	3	4	2	4	2	28	4
SA3	2	3	2	2	2	3	3	3	20	2

Table 5: Overview of results with suggested utilization.

Algorithm	Running time	Memory usage	Solution quality	Suggested utilization
DRPA-star	High	High	Optimal	Static-optimal quality
LMM	Low	Low	Low	Fast-low quality
GMM	Low	Low	Medium	Fast-high quality
SA1	Medium	Medium	High	Dynamic-medium quality
SA2	Low	Medium	High	Fast/Dynamic-high quality
SA3	Low	Medium	Very high	Dynamic-very high quality

To summarize the results and effectively identify the type of algorithm for each scenario, we will present an overview of our findings. As mentioned in the introductory passage, the main purpose of this paper is to present the vendor(s) or users with various types of placement algorithms, some sophisticated, some simple in nature and complexity. The overall winner in all the above algorithms in terms of solution, space and termination time was SA3. SA3 finished fourth on termination time, but its solution quality was not off by more than 10% in any of the experimental setups that were performed. Table 4 shows the numerical ranking of the algorithms based on the solution quality. Our recommendations are summarized in Table 5.

5 CONCLUDING REMARKS

In this paper we address the fine-grained replication of data among a set of sites in a distributed system such as the Internet and developed its cost model. We proposed five replica placement techniques based on the A-star algorithm. We also proposed two natural selection algorithms the LMM and the GMM. All the above proposed algorithms were compared and analyzed to identify techniques that can achieve optimal or fast or both types of solution. For a static system DRPA-star would be the best choice. For dynamic systems which require frequent updates, LMM could be used if time is the only constraint. But if both time and solution are important, then either of the sub-optimal A-star based heuristics can be used. The above recommendations are based on our comprehensive test analysis.

6 REFERECES

[1] T. Abdelzaher and N. Bhatti, "Web content adaptation to improve sever workload behavior," *Computer Networks*, 21(11), pp. 1536-1577, 1999.

[2] Y. Amir, *Replication using Group Communication over a Partitioned Network,* PhD dissertation, Hebrew University, Jerusalem, Israel, 1995.

[3] 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.

[4] R. Bunt, D. Eager, G. Oster, and C. Williamson, "Achieving load balance and effective caching in clustered web servers," in *4th International Web Caching Workshop*, pp. 159-169, 1999.

[5] K. Calvert, M. Doar, E. Zegura, "Modeling Internet topology," *IEEE Communications*, vol. 35, no. 6, pp. 160-163, 1997.

[6] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Transactions on Networking*, 9(4), pp. 253-285, 2001.

[7] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous computing systems," *IEEE Concurrency*, 6(3), pp. 42-51, 1998.

[8] J. Kangasharju, J. Roberts and K. Ross, "Object replication strategies in content distribution networks," in *Proc. of WCCD*, pp. 455-466, 2001.

[9] T. Loukopoulos and I. Ahmad, "Static and adaptive data replication algorithms for fast information access in large distributed systems," in *Proc. of ICDCS*, pp. 385-392, 2000.

[10] T. Loukopoulos, D. Papadias, and I. Ahmad, "An overview of data replication on the internet," in *Proc. of ISPAN*, pp. 31-36, 2002.

[11] M. Rabinovich, "Issues in web content replication," *Data Engineering Bulletin*, 21(4), pp. 21-29, 1998.