Replication in Overlay Networks: A Multi-objective Optimization Approach

Osama Al-Haj Hassan, Lakshmish Ramaswamy, John Miller, Khaled Rasheed, and E. Rodney Canfield

Computer Science Department, University of Georgia,
Athens, GA 30602, USA
{hasan,laks,jam,khaled,erc}@cs.uga.edu

Abstract. Recently, overlay network-based collaborative applications such as instant messaging, content sharing, and Internet telephony are becoming increasingly popular. Many of these applications rely upon data-replication to achieve better performance, scalability, and reliability. However, replication entails various costs such as storage for holding replicas and communication overheads for ensuring replica consistency. While simple rule-of-thumb strategies are popular for managing the cost-benefit tradeoffs of replication, they cannot ensure optimal resource utilization. This paper explores a multi-objective optimization approach for replica management, which is unique in the sense that we view the various factors influencing replication decisions such as access latency, storage costs, and data availability as objectives, and not as constraints. This enables us to search for solutions that yield close to optimal values for these parameters. We propose two novel algorithms, namely multi-objective Evolutionary (MOE) algorithm and multi-objective Randomized Greedy (MORG) algorithm for deciding the number of replicas as well as their placement within the overlay. While MOE yields higher quality solutions, MORG is better in terms of computational efficiency. The paper reports a series of experiments that demonstrate the effectiveness of the proposed algorithms.

Keywords: Replication, Multi-Objective Optimization, Evolutionary Algorithms, Greedy Approach.

1 Introduction

Overlay networks have evolved as scalable and cost-effective platforms for hosting several collaborative applications. Examples of overlay-based collaborative applications include instant messaging [1], content sharing [2] and Internet telephony [3]. However, the very fact that most of the overlays are formed from personal computers rather than powerful servers implies that collaborative applications running on top of them have to constantly deal with a variety of resource limitations such as storage and bandwidth constraints. Furthermore, the overlay networks experience significant *churn* with end-hosts constantly entering and exiting the network.

For collaborative applications to yield acceptable quality of service (qos), it is essential that the individual nodes of the system are able to access data-items in an efficient, scalable, and reliable manner. Replication of data-item is known to be an effective strategy for achieving better performance, scalability, and availability [4], and it has been utilized in a number of applications. However, data replication does not come for free; it consumes various resources like storage and network bandwidth. Replication imposes additional storage costs, and these costs are especially high in environments comprising of memory-scarce devices such as PDAs and cell phones [5]. Similarly, ensuring that replicas are consistent imposes communication overheads [6]. Thus, designing replication strategies involves balancing a variety of tradeoffs.

Two important questions in replicating data in overlay networks are: *(1) How many replicas of each data item should be maintained within the overlay?*; and *(2) Where (on which nodes of the overlay) should these replicas be placed?*. These two related challenges are collectively referred to as the *replica-placement problem*. Although replica placement in overlay networks has been previously studied [7][8][9][10][11], very few of the existing strategies take a holistic view of the various costs and benefits of replication. Many of them are limited by the fact that they consider only a small number of performance parameters. Even the ones that are sensitive to larger sets of performance factors, use simple rule-of-thumb strategies to manage the various tradeoffs. These schemes fail to optimally utilize the various resources available in the overlay.

In this paper, we propose a multi-objective optimization framework for the overlay replica placement problem. Our framework is characterized by several unique features. First, it takes into account several factors such as access latency, storage costs and availability. Second, these factors are regarded as *objectives for optimization* rather than constraints. This provides us with the advantage that we can *search* for solutions that yield close to optimal values for these parameters instead of just attempting to keep them within certain bounds. Third, our framework is inspired by evolutionary computing paradigm, and each solution is represented as a chromosome.

As a part of this framework, we propose two algorithms, namely, *multi-objective Evolutionary (MOE)* algorithm and *multi-objective Randomized Greedy (MORG)* algorithm. MOE algorithm is based upon the NSGA-II algorithm [13]. While MOE yields higher quality solutions, it is computationally intensive. To mitigate this concern, we propose the novel MORG algorithm, which is not only very efficient, but also yields solutions that are comparable in quality to those produced by the MOE algorithm.

We have performed series of experiments to study the costs and benefits of the proposed techniques. The results demonstrate the effectiveness of our multi-objective optimization framework in determining the locations for placing the replicas.

2 Background and Related Work

Replication techniques for overlays have been an active area of research. master-slave replication model has been adopted by several systems. For example, Sacha and Dowling [7] utilize a gradient technique for designing a master-slave replication for peer-to-peer networks. Replica placement problem has also received considerable research attention in recent years. In the Gnutella P2P system [2], only the nodes which request and retrieve a data-item hold a copy of that object. Freenet [8], however, allows replicas to be created on nodes other than the ones that requested the data-item. Cohen and Shenker [9] propose the path replication scheme, wherein replicas are placed along the path traversed by a data request. They also discuss random placement scheme where the replicas are placed on randomly chosen nodes in the overlay network. Some researchers have also addressed the question of how many replicas of a given data-item need to be maintained in the overlay. Proportional replication, wherein the number of replicas of a data-item increase with its usage has been discussed by Lv et al. [10]. They also contrast this scheme with a uniform replication scheme where the same number of replicas is created for all data-items. Benayoune and Lancieri [11] survey several replication techniques including the idea of replicating references of data-items instead of the data-items themselves.

Some of the existing techniques optimize the number of created replicas [16], while others optimize the locations in which to place replicas [17]; still others optimize how often replicas should be updated [18]. However, many of these techniques have the shortcoming that they only consider a limited set of parameters affecting the replication decision. The works by Loukopoulos and Ahmad [12] has the same objective as ours. They design a genetic algorithm to find the optimal replication strategy. In that work, two versions of the algorithm, a static version and a dynamic adaptive version are proposed. However, they model the problem as a single objective optimization problem. Specifically, they optimize latency, while storage, bandwidth and other parameters are considered as constraints. One of the limitations of this approach is that it can only maintain the constraint parameters within certain bounds, but cannot explicitly optimize them. Further, their work did not take into account the reliability of the system. Thus, we believe that there is a need for a holistic approach to the overlay replica placement problem which not only takes all the important factors into account but also explicitly optimizes them. Motivated by this need, we propose the MOE and MORG algorithms, both of which are based upon the multi-objective optimization paradigm.

3 Architectural Overview

Our system is based upon unstructured P2P overlays. Unlike their structured counterparts, unstructured overlays do not provide distributed hash table (DHT) support. In these networks, content searching happens purely by ad-hoc messaging among neighbors. In our architecture, a data-item and its replicas are viewed

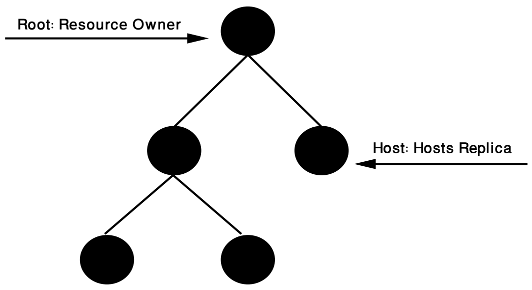


Fig. 1. Logical tree of nodes hosting replicas

as a logical tree. We build a replication tree for each data-item in the system in which the root would be the owner of the original copy of the data-item , and the other nodes of the tree would have replicas. Fig. 1 illustrates the replication tree. The replication tree is constructed using a scheme similar to the one proposed by Zhang et al. [15].

When a node in the system wants to *read* a data-item it accesses the closest replica of the data-item. However, updates of a data-item are always initiated at root of the tree. A node that wants to update a data-item, sends it to the root, which is then propagated down the tree-hierarchy. The root node as well as the other nodes holding replicas collect various statistics such as the frequency at which a replica is used, frequency at which the data-item is updated, ratio of reads to writes for a data item, failure statistics of the node holding the replica, and storage availability and utilization at each node. The information collected at various nodes will be aggregated at the root of the tree. These statistics will be fed into our optimization engine which produces solutions to the replica placement problem, indicating which nodes should hold additional replicas and which of the existing replicas need to be de-commissioned. The replication tree is then modified accordingly. Fig. 2 illustrates the functioning of our system.

4 Problem Formulation

The problem on hand consists of optimizing a set of objectives, some of which might be conflicting with one another. For example, achieving better latency might require creating additional replicas. However, doing so would invariably increase storage and consistency maintenance costs. Similarly, placing the replicas on most stable nodes might not be ideal from latency minimization perspective. Multi-objective optimization deals with these conflicting objectives by evolving a set of solutions that compromise these conflicting objectives.

The quality of solutions obtained by multi-objective optimization is inherently dependent upon how well the objectives are formulated. In this section, we first model each of the objectives following which we discuss their conflicts.

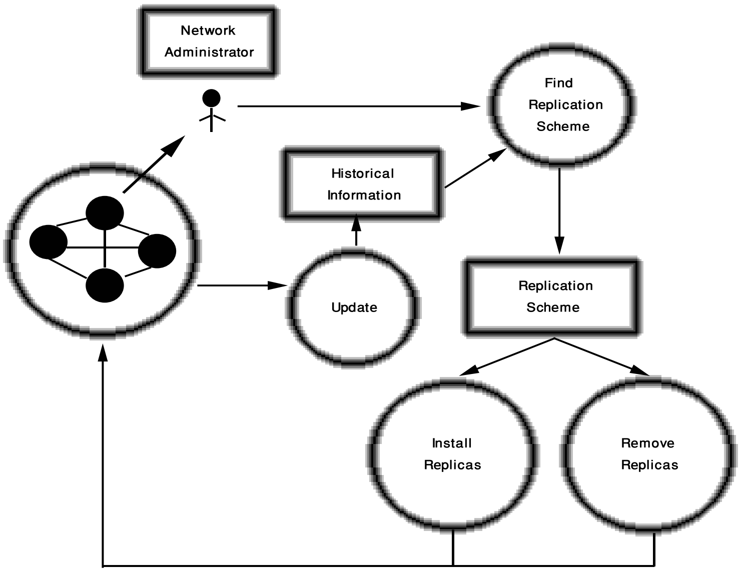


Fig. 2. System functionality overview

4.1 Latency

Minimizing latency is important for any collaborative system. Minimizing latency depends on utilizing high bandwidth channels, as high bandwidth channels yields lesser latency. Thus, it is better to avoid nodes with low-bandwidth connections. Furthermore, if a node frequently accesses a data-item, then minimizing its latency in retrieving the data-item should naturally take priority over those nodes that occasionally access the same data-item. Taking these aspects into account, we model the latency objective function D as the following:

$$D = R + W \tag{1}$$

where

$$R = \sum_{i=1}^{n}\sum_{j=1}^{m} E(i,s_j) * \frac{Z(s_j)}{B(i,H(s_j))} * RP(i,s_j) \tag{2}$$

and

$$W = \sum_{i=1}^{n}\sum_{j=1}^{m}(1-E(i,s_j)) * \frac{Z(us_j)}{B(i,O(s_j)))} * WP(i,s_j) + \sum_{k=1}^{x} \frac{Z(us_j)}{B(O(s_j),k)} * WP(i,s_j) \tag{3}$$

Where
R: Total read cost in the system
W: Total write cost in the system
i: index for nodes of the system

j: index for data-items of the system
x: index for nodes holding replicas of data-item j
s_j: is the data-item for which we are trying to find read cost.
E(i,s_j): equals to 0 if data-item j exists on host i, otherwise it equals to 1
H(s_j): is the machine that hosts replica of data-item j
RP(i, s_j): is the percentage of read requests coming from node i asking for data-item j
WP(i, s_j): is the percentage of write requests coming from node i updating data-item j
us_j: is the updated version of data-item j
O(s_j): is the owner of data-item j
B(a,b): Minimum bandwidth along the path from node a to node b
Z(s): is the size of the data-item 's'

We assume that the system is fully active, which means that each node in the system attempts read and write requests, each node according to its read requests percentage (RP) and write requests percentage (WP). Basically, what happens in a read operation is that a data item needs to be transferred in chunks to the requester; this is determined by dividing the size of the data item on the minimum bandwidth along the path from source to requester. With regard to write requests, the first part of the equation before the plus sign models transferring the data item in chunks to its master copy owner and the second part of the equation after the plus sign models the propagation of the data item in chunks from the master copy owner to all nodes holding a replica of the data item.

4.2 System Reliability

Network churn is an inherent characteristic of a P2P overlay. While nodes continuously enter and exit the network, some nodes are more stable than others. For example, machines that are connected through wireless links are more likely to disconnect from the network than those that are on wired connections. Naturally, it is preferable to place replicas on more stable nodes. In our system, replica reliability is expressed through the failure probability of the node hosting it, and reliability maximization is achieved by minimizing replica failure probability. Hosts failure probabilities are drawn randomly and they can be updated by tracking the history of hosts in the system. Reliability objective function SR is modeled through the following function:

$$SR = \prod_{i=1}^{n} \prod_{j=1}^{m} E(i, s_j) * F(i) \tag{4}$$

Where
i: index for nodes of the system
j: index for data-items of the system
E(i,s_j): equals to 0 if data-item j exists on host i, otherwise it equals to 1
F(i): failure probability of node i

4.3 Storage

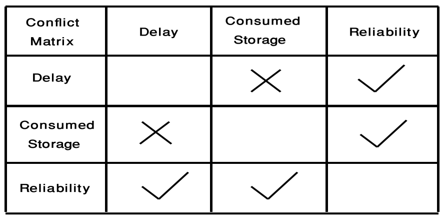
In a heterogeneous network, nodes can have different storage capacity constraints. Usually each node has an upper limit on the storage that can be utilized by the overlay applications. This makes our objective here is to minimize storage consumption on each node of the overlay taking into consideration the total storage available at the node. Storage objective function SC is given by the following:

$$SC = \sum_{i=1}^{n} SC(i) + \sum_{j=1}^{m} (1 - E(i, s_j)) * Z(s_j) \tag{5}$$

Where
i: index for nodes of the system
j: index for data-items of the system
SC(i): is the storage consumption on node i.
E(i,s_j): equals to 0 if replica j exists on host i, otherwise it equals to 1
Z(s): is the size of the data-item 's'

4.4 Conflicting Objectives

Multi-objective optimization is most appropriate when the objectives conflict. If there are no conflicts, we will end up with one solution. Conflict among objectives results in a set of compromise solutions. Basically, the conflict in our system occurs whenever we have good values for one objective and bad values for another objective. If we have good values in both objectives or bad values in both objectives, then these are the best case and the worst case scenarios respectively, which do not usually exist in overlay networks. Two types of conflicts exist in our system. The first conflict is latency-reliability conflict. The second conflict is storage-reliability conflict. The former conflict occurs because as more replicas are installed in the system, latency tends to increase and reliability tends to increase. Latency tends to increase as more replicas are installed in the system because the propagation update cost resulted by write requests tends to overwhelm read cost savings resulted by read requests. For sure, reliability increases as more replicas are installed in the system because whenever a failure occurs, the system functionality is preserved because of the replicas we have. Fig. 3 shows conflicts between objectives. The cross mark indicates that a conflict does not exist and a tick mark indicates that a conflict exists.



Conflict Matrix	Delay	Consumed Storage	Reliability
Delay		✗	✓
Consumed Storage	✗		✓
Reliability	✓	✓	

Fig. 3. Conflicts between objectives

	N1			N2			···			Nn	
S1	···	Sm	S1	···	Sm	···	···	···	S1	···	Sm
0	···	1	0	···	0	···	···	···	1	···	1

Fig. 4. Binary representation of a replication scheme

5 Our Approach: Multi-objective Optimization

Our approach depends on taking the historical system information and feed it to an engine where we try not only to keep latency, reliability, and storage within constraints, but in addition we try to optimize latency, reliability, and storage in order to find different trade offs between these objectives. Since we are using more than one objective, we are doing multi-objective optimization.

5.1 Solution (Chromosome) Representation

A solution in our system is a combination of nodes that will hold replicas of a given data-item. We use a binary representation in which a value of 1 means hosting a replica and the value of zero means no hosting of a replica. Fig. 4 shows the binary representation of one chromosome for a system of n nodes and m data-items. The first row has nodes labels, the second row has data-items labels, and the third row has the general binary representation of the chromosome.

5.2 Multi-objective Evolutionary (MOE) Optimization

In this technique, we apply multi-objective optimization to the problem at hand. Specifically, we use an existing algorithm called NSGA-II [13]. Multi-objective optimization is one of several techniques in evolutionary computing. Evolutionary computing is the branch of science that takes randomness as a mean of problem solving; it also considers solutions of the problem as chromosomes. Mating between different chromosomes could yield a better breed or better solutions. Using evolutionary computing techniques is very helpful in situation where the search space of a problem is huge; searching this huge space in sequential search techniques takes exponential times. Evolutionary computing jumps in the search space in such away that explores areas in which a potential good solution can be found. Many of evolutionary computing techniques rely on operators such as crossover operator which is used for mating between chromosomes, mutation operator which is used to alter genes of the chromosomes, parent selection operators which is responsible of choosing chromosomes for mating. Doing the mating process continues over and over until specific conditions are met such as accuracy of solution or no change over the best solution.

NSGA-II Multi-objective Optimization. This algorithm is a low computation, elitist approach, parameter-less niching, and simple constraint handling strategy algorithm. A non-dominated based sorting technique is used in the algorithm. Furthermore, a selection operator that selects parents based on fitness and spread of mating pool members is adopted. Having solutions of the replication problem as chromosomes, the algorithm selects solution for mating from a set of previously initialized solution set, the chosen solutions mate together and produce more solutions, the new solutions are added to the solution set, the solution set is cut to fronts based on comparisons between solutions, a solution is better than another if it dominates the other solutions. Solution 'A' dominates solution 'B' if solution 'A' is better or equal to solution 'B' in terms of all criterions, namely, delay, reliability, and consumed storage. For our system, we care about the first front which contains the set of solutions that are not dominated by any other solutions. This process continues until a specific number of fitness evaluations are attempted. It can also be set to continuous execution until a specific time expires or until there is no further improvement.

The core part of NSGA-II algorithm is listed below. The intuition behind using this algorithm is the ability to find several fronts of solutions using ranking and crowding. This will give us a variety of solutions, which is done in lines 6,7, and 8 of the while loop. In our experiments we used binary representation of solutions, binary tournament selection, single point crossover for mating with a probability of 0.9 to perform crossover and bit flip mutation for mutation operation with a probability of (1/number of nodes) to make mutation.

The complexity of this algorithm is O(MN^2). Where M is number of objectives and N is the population size.

Chore loop of NSGA-II multi-objective evolutionary approach

```
set initial population size pSize
set maximum number of evaluations maxEval
for ( i iterations from 1 to pSize)
  initialize a solution Sol i
  calculate Soli fitness
  add soli to population pool
  evaluations = evaluations + 1
end-for
while (evaluations < maxEvaluations)
  select parent P1 from population pool
  select parent P2 from population pool
  perform crossover between P1 and P2 and get child C1,C2
  evaluate C1 fitness and C2 fitness
  add C1, C2 to population pool
  perform ranking on population pool
  assign crowding distance to individuals of population
  get the front of individuals of the population pool
  add the front to the solution set solSet
  evaluations = evaluations + 2
end-while
return solSet
```

(Chore loop of NSGA-II multi-objective evolutionary approach)

5.3 Multi-objective Randomized Greedy (MORG) Optimization

One of the characteristics of the evolutionary multi-objective approach is that it takes significant time to converge. So, we need algorithms that can converge in reasonable amount of time. In general, the greedy algorithms are good candidates when it comes to fast execution. Ordinary greedy algorithms do not use dominance as criteria of deciding the best among individuals. So, we use dominance to drive the greedy decisions. Also, greedy algorithms generate one solution. But, in our system, the notion of conflicts between objectives implies that there is no one best solution, so we use a multiple random starting points to generate different solutions. A pseudo code is listed below for the multi-objective randomized greedy approach. The intuition behind using this algorithm is its fast execution time which could be necessary sometimes if we need to generate quick solutions when the network is not in a good shape. The uniqueness of this algorithm relies in the several starting points that give us a variety of solutions and relies in the use of dominance as criteria of comparing solutions. The first for loop of the algorithm generates different starting points to generate solutions from. Each starting point is basically a node in the system. In each starting point, the algorithm considers replicating on neighbors. Here, two cases may happen, if replicating on any of the neighbors does not dominate the current solution, then the current solution is the final solution. Otherwise, we replicate on one of the neighbors which dominate the current solution and the process will be repeated from that neighbor until no further improvement can be found.

Multi-objective randomized greedy approach

```
set number of desired solutions solDesired
for (n iterations from 1 to solDesired)
  current node currNode = random node
  initialize current solution currSoln by having 1's on
   data-item owners positions.
  for(i iterations from 1 to number of data-items)
// Tests if replicating data-item i on currNode is better
 than current solution
    testResult = dominanceTest(currSoln,currNode,i)
    if  (testResult = 1)
add replicating data-item i on node currNode to currSoln
    end-if
    progress=true
    while (progress=true)
progress=false
neb = neighbor list of currNode
solk is the solution coming from replicating data-item i
 on node nebk
      // Tests if replicating data-item i on currNode is
       better than current solution
      testResult = dominanceTest(currSoln,nebk,i)
      if (testResult =  1)
        save the best nebk of the neighbors in terms of
         dominance
        progress = ture
end-if
    end-while
  end-for
  add soln to set of final solution solSet
end-for
return solSet
```

(Multi-objective randomized greedy approach)

6 Experiments and Results

6.1 Experimental Setup

The code we used for NSGA-II implementation is available on jMetal [14] which is an object-oriented java-based framework that eases the development, testing, and working with metaheuristics for solving multi-objective optimization problems (MOPs). Our experiments were done over unstructured peer-to-peer overlay with 50, 75 and 100 nodes and 1 data-item. Experiments can be extended to incorporate larger numbers of data-items. In each experiment we study the trade-off between two factors, because it is difficult to clearly visualize the results if several factors are simultaneously varied in a single experiment.

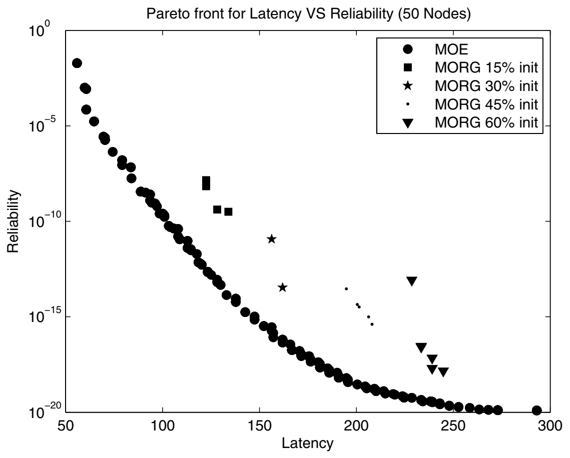


Fig. 5. Pareto front of latency VS reliability (50 Nodes)

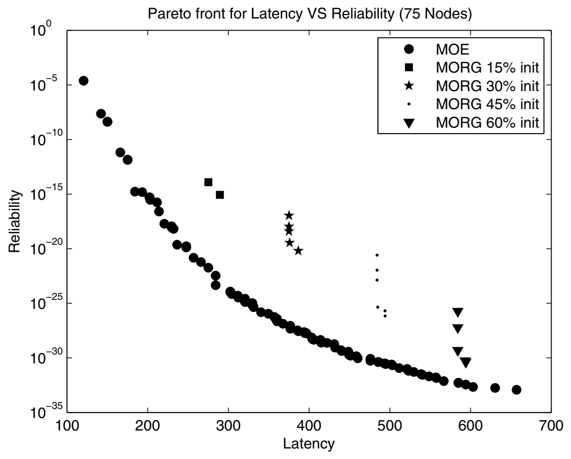


Fig. 6. Pareto front of latency VS reliability (75 Nodes)

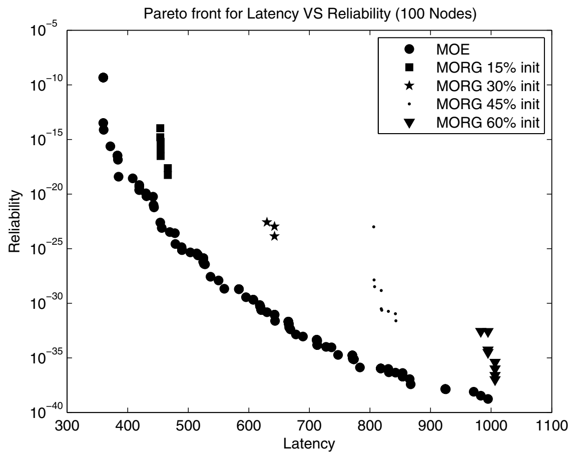


Fig. 7. Pareto front of latency VS reliability (100 Nodes)

6.2 Latency-Reliability Tradeoff

Fig. 5, 6 and 7 show the pareto front of latency against reliability in experiments that involved 50, 75, 100 nodes respectively. latency values are better when they are low, and because reliability is expressed in terms of node failures, lower failure values mean better reliability. It is clear how the MOE approach gives a variety of solutions that include a trade off between latency and reliability. The system administrator can choose a solution based on the network status. For example, if the network is not reliable because of nodes departures, he can pick a replication scheme that increases network reliability. If the network suffers from high latencies, a replication scheme with low latency is good. The MORG approach gives good values based on the percentage of nodes that get initialized with replicas in advance of running the algorithm, but still, it does not give as good values as MOE approach. It is good to use MORG when the network is in a critical condition that needs a quick solution.

6.3 Storage-Reliability Tradeoff

Fig. 8, 9 and 10 show the pareto front of available storage against reliability in experiments that involved 50, 75, 100 nodes respectively. Available storage

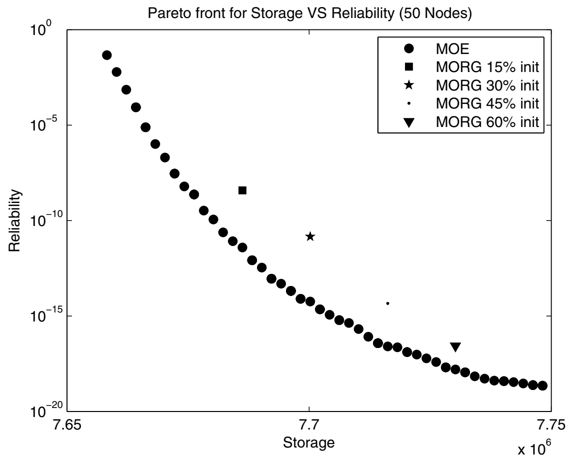


Fig. 8. Pareto front of storage VS reliability (50 Nodes)

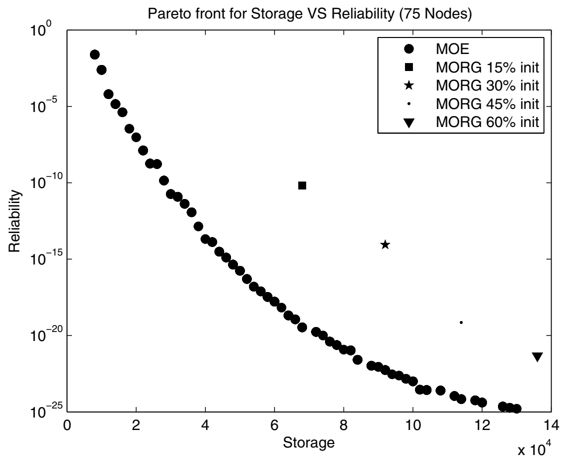


Fig. 9. Pareto front of storage VS reliability (75 Nodes)

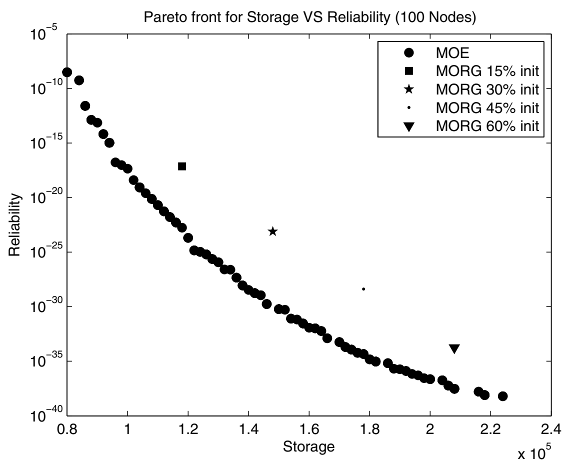


Fig. 10. Pareto front of storage VS reliability (100 Nodes)

values are better when they are high, reliability are expressed in terms of node failures, so lower failure values means better reliability. It is clear how the MOE approach gives a variety of solutions that include a trade off between available storage and reliability. The system administrator can choose a solution based on the network status. For example, if the network is not reliable because of nodes departures, he can pick a replication scheme that increases network reliability. If the network nodes suffer from lack of storage, then the priority changes to selecting a replication scheme with high available storage. The MORG approach

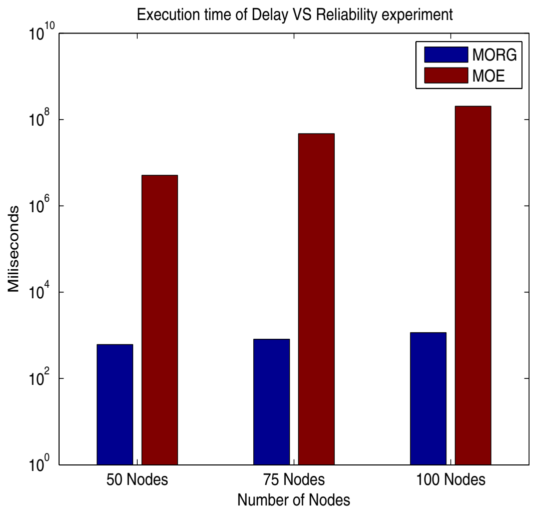


Fig. 11. Execution time for MOE and MORG in a latency vs reliability experiment

gives good values based on the percentage of nodes that get initialized with replicas in advance of running the algorithm, but still, it does not give as good values as MOE approach. It is good to use MORG when the network is in a critical condition that needs a quick solution.

6.4 Execution Time

Fig. 11 shows the execution time for the 2 algorithms in the 3 experiments. The downside for the multi-objective evolutionary approach is the long time it needs to give us good results; this is why we choose to run it overnight when the network activity is low. Nevertheless, MOE gives better results than MORG approach with respect to execution time.

7 Discussion

In [12], the authors deal with the replication problem as a single objective optimization; they optimize latency taking into consideration satisfying constraints related to some parameters like storage availability. Dealing with these parameters as constraints will ensure that the constraints are met. But they did not find the best value for those parameters. This is not the right approach. A justification of that is the following. In many cases, nodes of the system are not dedicated to a specific service, and they might host different services. Those services might consume nodes resources such as storage, each node according to its behavior. At some point, the system could suffer from lack of storage, which means that storage in this case is a vital resource, and we should maximize it as much as possible instead of keeping it under a certain level using a constraint. This is why the multi-objective optimization approach is better than the single objective genetic algorithm approach. If we are in a situation where the network is in a critical condition that requires a quick solution, then the best approach to be used is MORG. Sometimes, a system can have low traffic during certain overnight hours, and in this case the best approach to choose is MOE because of the wide variety of solutions that give us the best optimization for our objectives. Since the multi-objective evolutionary approach takes long execution time, one thing to be done is to use forecasting techniques that help us to estimate a good time to execute the algorithm instead of executing the algorithm regularly. Also, in cases where the system administrator is monitoring the network, he can simply execute the algorithm whenever he finds a necessity to do so and this can minimize the number of times in which the multi-objective evolutionary algorithm needs to be executed.

8 Conclusion

While many overlay-based collaborative applications rely upon data-replication for achieving better scalability and performance, data replication also involves various overheads. Replica placement is one of the key problems in overlay-based replication schemes. This paper proposes a novel multi-objective optimization approach for replica-placement in an overlay. One of the key strengths of our approach is that we view various factors influencing replication decisions such as access latency, storage costs, and data availability as objectives, and not as constraints, which allows us to search for solutions that optimize these parameters. Specifically, we propose two multi-objective optimization algorithms. The multi-objective evolutionary (MOE) algorithm is based on the NSGA-II algorithm, and it has the advantage of providing us with very high quality solutions albeit a longer execution time. On the other hand, multi-objective randomized greedy (MORG) algorithm is characterized by its superior computational efficiency, and it yields solutions that are of comparable quality. We report several experiments to study the effectiveness and performance of the proposed algorithms.

References

1. Minar, N., Hedlund, M., Shirky, C., O'Reilly, T., Bricklin, D., Anderson, D., et al.: Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly Media Inc., Sebastopol (2001)
2. Gnutella Protocol Specification, `www9.limewire.com/developer/gnutella_protocol_0.4.pdf`
3. Baset, S., Schulzrinne, H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In: 25th IEEE International Conference on Computer Communications, Spain, pp. 1–11 (2006)
4. Yu, H., Vahdat, A.: The Costs and Limits of Availability for Replicated Services. In: 18th ACM symposium on Operating systems principles, Canada, pp. 29–42 (2001)
5. Teuhola, J.: Deferred maintenance of replicated objects in single-site databases. In: 7th International Workshop on Database and Expert Systems Applications, Finland, p. 476 (1996)
6. Chun, B., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M.F., Kubiatowicz, J., Morris, R.: Efficient replica maintenance for distributed storage systems. In: 3rd Symposium on Networked Systems Design and Implementation, California, p. 4 (2006)
7. Sacha, J., Dowling, J.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. In: Databases, Information Systems, and Peer-to-Peer Computing, International Workshops, pp. 181–184. IEEE Press, New York (2005)
8. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: ICSI Workshop on Design Issues in Anonymity and Unobservability, pp. 181–184. IEEE Press, California (2000)
9. Cohen, E., Shenker, S.: Replication Strategies in Unstructured Peer-to- Peer Networks. In: ACM SIGCOMM Computer Communication, pp. 181–184. IEEE Press, New York (2002)
10. Lv, Q., Cao, E., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer networks. In: 16th ACM International Conference on Supercomputing, pp. 181–184. IEEE Press, New York (2002)
11. Benayoune, F., Lancieri, L.: Models of Cooperation in Peer-to-Peer Networks, A Survey. In: 3rd European Conference on Universal Multiservice Networks, pp. 181–184. IEEE Press, New York (2004)

12. Loukopoulos, T., Ahmad, I.: Static and Adaptive Distributed Data Replication using Genetic Algorithms. Journal of Parallel and Distributed Computing 64, 1270–1285 (2004)
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm NSGA-II. IEEE transactions on evolutionary computation 64, 182–197 (2002)
14. Metaheuristic Algorithms in Java, `http://mallba10.lcc.uma.es/wiki/index.php/JMetal`
15. Zhang, J., Liu, L., Ramaswamy, L., Zhang, G., Pu, C.: A Utility-Aware Middleware Architecture for Decentralized Group Communication Applications. In: ACM/IFIP/USENIX Middleware Conference. New port beach, California (2007)
16. Chen, Y., Katz, R.H., Kubiatowicz, J.D.: Dynamic Replica Placement for Scalable Content Delivery. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 306–318. Springer, Heidelberg (2002)
17. Mansouri, Y., Monsefi, R.: Optimal Number of Replicas with QoS Assurance in Data Grid Environment. In: Second Asia International Conference on Modelling and Simulation, Kuala Lumpur, pp. 168–173 (2008)
18. Tu, M., Tadayon, T., Xia, Z., Lu, E.: A Secure and Scalable Update Protocol for P2P Data Grids. In: 10th IEEE High Assurance Systems Engineering Symposium, Texas, pp. 423–424 (2007)