Locality Based Protocol for MultiWriter Replication systems

Lei Gao

Department of Computer Science

The University of Texas at Austin

lgao@cs.utexas.edu

One of the challenging problems in building replication systems is how to design the appropriate replication protocol set that can achieve the target level of availability, performance, and consistency while minimizing the replication cost. But various existing replication protocols could not provide the scalability and performance desired by replicated applications when the workload characteristics change dynamically. We propose a locality based replication protocol that combines the benefits of existing protocols. By taking advantage of the read/write locality, this protocol scales well under workload with all read-to-write ratios.

1 Introduction

One of the challenging problems in building replication systems is how to design the appropriate replication protocol set that can achieve the required level of availability, performance, and consistency while minimizing the replication overhead at the specific replication scale. Replication techniques are primarily employed to improve the availability [5, 8, 10, 12] and the performance [3, 6] of distributed applications and web services deployed over wide area networks. However, in order to maintain a consistent view across all replicas in the system, additional workload may be imposed on the system upon on both client reads and writes. As the result, the replication systems usually gain better availability and performance at the cost of reducing the overall system capacity.

Research work has investigated the scalability of various replication protocols under different

workload [7]. The result shows that some protocols scale well when reads dominant in the overall workload while some other protocols can handle write-concentrated workload relatively better. For example, the *write-all-read-any-available* protocol, ROWAA, scales well against workload with high read ratio. But as the write ratio increases, this protocol incur high cost because the cost of write operations is proportion to the replication scale. On the other hand, the grid quorum protocol scales better for high write ratio workload because the write cost is only square root of the overall replication size.

The goal of this project is to understand the workload impact to the scalability of existing replication protocols and to design a new protocol to scale under workload with all read-to-write ratios. In particular, we exploit the impact of the read and write locality resulted in the access routing in the replication system. Strong access locality is the key for our protocol to optimize its scalability and response time. We introduce invalidations in the grid quorum protocol to reduce the read cost when the read locality is considered. Furthermore, subsequent writes to the same write quorum can avoid invaliding non-write-quorum servers which are already invalidated by the first write.

The significance of our locality based replication protocol is that the protocol scales independently of the read-to-write ratio. Because the workload characteristics is hard to predict and changes dynamically at runtime, building a replication system that scales well under the runtime environment involves sophisticated algorithms for analyzing, predicting, and adapting to the actual workload. This approach increases the complexity of the development. The alternative approach is to manually reconfigure the protocol to accommodate the workload changes at runtime. However, this approach suffers from high maintainness cost and relatively in-responsive. Our locality based replication protocol takes advantage of access locality to keep a low replication cost for both read and write operations. This protocol provides a straightforward solution for developing highly scalable and efficient replication system.

The rest of the report is organized as follow. Section 2 puts this report into context by discussing some representing protocols that are commonly used for building replication systems and some other related work. Section 3 presents the benefits and limitations of two replication protocols that are leveraged by our locality based protocol. Section 4 explain the design of the locality based protocol, followed by Section 5, the protocol evaluation. And Section 6 concludes the project.

2 Related

Some representative replication protocols include ROWAA, and quorum protocols. The major difference among those protocols is the number of replicas contacted for every read or write operation.

The ROWAA protocol, proposed by Bernstein [2], ensures to propagate writes to all the replicas in the system to guarantee that reads only have to read from one replica.

Comparing with ROWAA, the quorum protocols reduce the cost of performing a write in the system. Instead of forwarding writes to all replicas, the protocol chooses to forwarding writes a quorum of replicas and multicast reads to a read quorum. The quorum protocol guarantee that a read quorum intersects with a write quorum to allow reads to return the fresh value. The size of the quorum affects the scalability and performance of the specific protocols. Various work propose different techniques for constructing quorum. In grid quorum [4], replicas are organized in a grid of rows and columns. Writes are performed to a row while reads are performed to a column. In the majority quorum algorithm [11] read and write quorum mush fulfill the constrains: 1. write quorum size is at least one greater than half of the replication size; 2. the sum of write and read quorum sizes is larger than the replication size. In a tree quorum [1] the size of write quorum is reduced while the read quorum is one.

3 background

Our locality based replication protocol is evolved from two existing protocols, ROWAA and grid quorum protocol.

3.1 ROWAA

For the simplicity and driven by the characteristics of web client workload, the ROWAA protocol is widely used in many web service implementations. In ROWAA protocol, writes are simplify forwarded to all replicas. The the synchronous case, a write operation does not return until it hears responses from all replicas. And in the asynchronous case, a write is eventually forwarded to all available replicas. Because all replicas contain the data resulted in the last write, read operations

can be performed on any available replicas to retrieve the value. One advantage of ROWAA protocol is that its implementation is relatively straightforward. Multicast can be used to propagate writes and unicast and be used to access any random replica to retrieve the value. Another advantage of this protocol is that it scales well when handling workload dominated by reads. Since majority of today's web traffic is read, this protocol is preferable for building replicated web services. The disadvantage of ROWAA protocol comes from its write-to-all-available design. Clearly, in a large scale replicated system, when a client write operation need to be propagated to all available replicas, the overall system load increases as the number of replicas increase. The ROWAA protocol is scalable and responsive only under the read-dominate workload.

3.2 Grid

Grid Quorum protocol trades the read cost for less write cost. As described in the previous section, grid quorum forwards writes to a row of replicas instead of forwarding to all replicas in the system. The cost of write operations is reduced from n to $\sqrt{n}$. Meanwhile, the read cost is increased from 1 to $\sqrt{n}$ because reads have to query from a column of replicas to determine the most updated value. This protocol balances the cost between reads and writes. And because its reads and writes cost are both $sqrtn$, it provides consistent scalability for all workload. But the negative side is that reads, which are often the dominating web client workload, becomes more costly in terms of both overall system load and response time. In addition, organizing replicas as grid quorum for both reads and writes contributes much complexity in building replication systems.

4 Design

The goal of this project is to investigate some limitations of existing replication protocols. Then, we try to design a new protocol that combines the benefits and eliminates shortcomings of those existing protocols.

4.1 Invalidations

We use grid quorum as the base of our new protocol because it seems to provide a consistent scalability across workload with all read-to-write ratios. To further reduce the read cost, we introduce invalidations in the protocol. For every write initiated by client, the system not only forwards it to the a write quorum but also sends invalidations to all other replicas outside of this write quorum. Therefore, for a given replicated object in the system, any replica will contain the fresh state of this object, or otherwise the invalidation. In this case, we know that a particular replica contains the fresh state of an object if this replica returns a value. Otherwise, when a replica returns a invalid state, we can be sure that we need to query other replicas for the value. Despite of the fact that in the worst case a read still query a read quorum, $\sqrt{n}$ replicas, some reads can obtain the desired value by querying less than $\sqrt{n}$ replicas. Therefore, the average read cost is reduced to less than $\sqrt{n}$.

4.2 Locality

The cost of read and invalidation can be further reduced by using locality based routing in the replication system. When a client access the service of a replicated system, the client will be routed to one of the replicas based on specific routing algorithms. Such routing algorithms include load-balancing algorithm, locality based algorithm. For our target replication system, we assume that the locality based algorithm is used for client requests routing. In other words, our system tends to route both read and write requests of the same client to the same replica as long as this replica is available. There are two benefits for using such routing policy.

1. **Invalidation elimination.** When a particular client tries to modify an object, the write is originally sent to replica *RA*, then propagated to other replicas in a write quorum, *RB, RC* (assuming a 3x3 grid). Meanwhile, other replicas, *RD ... RI*, receive invalidations to this object. If this client tries to modify the same object again, the locality based routing policy will send the second write to the same replica *RA*. At this time, the protocol only propagates the second write to the same write quorum, *RB, RC* without re-invalidating other replicas, *RD ... RI*. Therefore, the cost for the second write is only $\sqrt{n}$ instead of n like the first write. All subsequent writes initiated by this client to the same object will be routed to *RA* as long

as this replicas is available and their cost will all be $\sqrt{n}$.

2. **ROWAA-alike read.** After modifying an object, a client may want to query the state of this object by sending a read request to the system. Similarly, this read request will be routed to *RA* first. This time, since *RA* contains the value resulted by last write, *RA* can immediately return this value to the client without having to query from other replicas in a read quorum to determine the fresh value. As long as the read requests of the same object from this client are routed to *RA*, the read cost is 1.

Occasionally, *RA* will be partitioned from the rest of replicas due to network failures so that this particular client's read/write requests are routed to other replicas, such as *RD* or *RH*, which are not in the same write quorum as *RA*. In this case, the cost of the first write routed to *RD* or *RH* is n because this write is propagated to another write quorum and the rest will receive invalidations. In another case when the first request arriving at *RD* or *RH* is a read request, this replica has to query a read quorum containing either *RB* or *RC* to retrieve the desired value. This read request incurs a cost of $\sqrt{n}$.

According to the previous analysis on Internet service availability, an average web client cannot contact an average web server for about 15 minutes a day [**?**, 9, 13]. It suggests that the probability for a client can be routed to the same replicas is roughly 99%. Therefore, the locality based replication protocol can provide reads at the cost similar to ROWAA and writes at the cost of $\sqrt{n}$ under normal conditions. However, the cost of both reads and writes can increase substantially as the network partition occurs more often under some circumstances. We will evaluate the affect of locality to the protocol scalability and responsiveness in the next section.

5 Evaluation

In this section, we seek to justify the tradeoffs among scalability, responsiveness, the read/write locality, the replication size, and the read-to-write ratio. First, we will describe the model that is constructed based on the locality based protocol. Then, we analyze the simulation results produced from this model and compare them to those of ROWAA and grid quorum protocols.

5.1 Modeling

Our model consists of three major components, the client workload generator, the replication cluster, and the replicated server.

5.1.1 Client workload generator

This component is responsible for generating the client workload that simulates the characteristics of the runtime client behavior. Client requests are generated from this component and become the input of the replication cluster eventually. The client workload generator specifics several requests' attributes during the generation phase. The workload characteristics includes the client ID, the request arrive rate, request types, e.g. read or write, and request origination, e.g. traveling or at-home. Those attributes are essential for requesting routing and processing by the replication system.

5.1.2 Replication cluster

The cluster receives requests from the generator and routes the them to the appropriate replicas based on the client IDs and requests' origination information. While processing the client requests, the replicas initiates additional requests to other replicas to propagate changes or query for fresh values. The replication cluster routes those replica-initiated requests based on the receivers' information embedded within requests. Before a request reaches a replica, the replication cluster randomly delays the request as if this request is traveling through a wide area network before reaching a replica. The cluster also returns the value to the client as soon as the value is obtained from the replica that this client is contacting.

Replicas exist in the model in the form of an array. Requests (originated from both clients and replicas) are routed to replicas by providing to the cluster an index representing the position of their target replica in the replica array. There is one delay unit associated with each replica so that the model can simulate the real world scenario where the propagation delays differ from between two replicas pairs or two client-replica pairs.

5.1.3 Replicated Server

When a replica receives a client read request, it returns the value to the client if the local value is not invalidated. Otherwise, this replica initiates server read requests and sends them to a quorum of replicas. Each replica in this read quorum response to the original replica with either the value or a invalid state. As soon as the original replica receives a response contain the actual value, it returns the value back to the client. When a replica receives a client write request, it propagates this write to a quorum of replicas. Before this replica sends out invalidations to other replicas outside the write quorum, it checks whether its local value is valid. The invalidations are only sent out when this replica does not have a valid previous value.

5.2 Simulations

In this subsection, we seek to understand the scalability and responsiveness of the locality based protocol in contrast with the ROWAA and grid quorum protocols. Notice that changes to any of the multiple parameters can alter the scalability and responsiveness of the protocol. Our experiments target the read-to-write ratio, replication size, and read/write locality.

5.2.1 Replication size

Graph 1 illustrates the effect on scalability as the replication size changes. This experiment is conducted with the workload consisting 50% writes. In this graph, the x-axis represents the replication size in terms of the number of replicas used in the system, and the y-axis represents the normalized overall system workload with respect to workload generated for each replication size. The top most curve represents the normalized overall system workload of ROWAA protocol, which shows competitive scalability at the small replication scale. As the replication scale increase, the high cost of ROWAA becomes obvious as its representing curve increase sharply. The locality based protocol shows the competitive scalability with grid quorum protocol when the locality is 90/grid quorum.

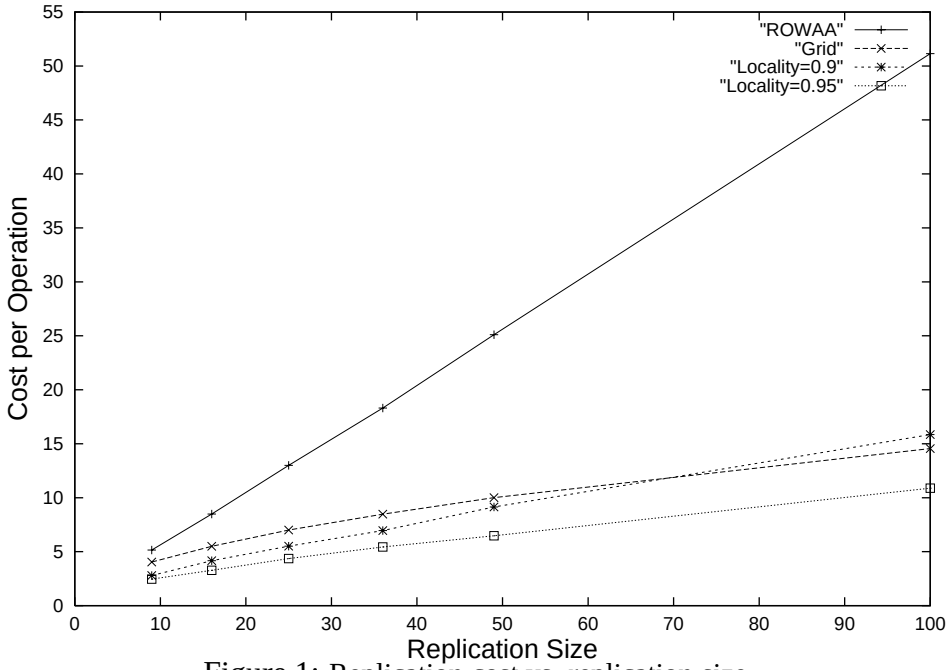

Figure 1: Replication cost vs. replication size.

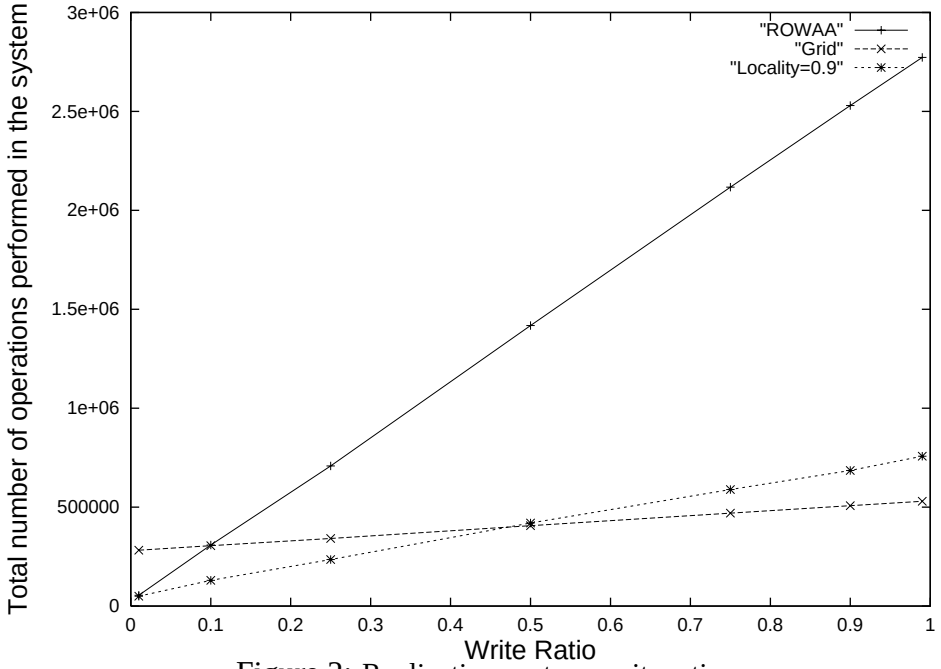

Figure 2: Replication cost vs. write ratio

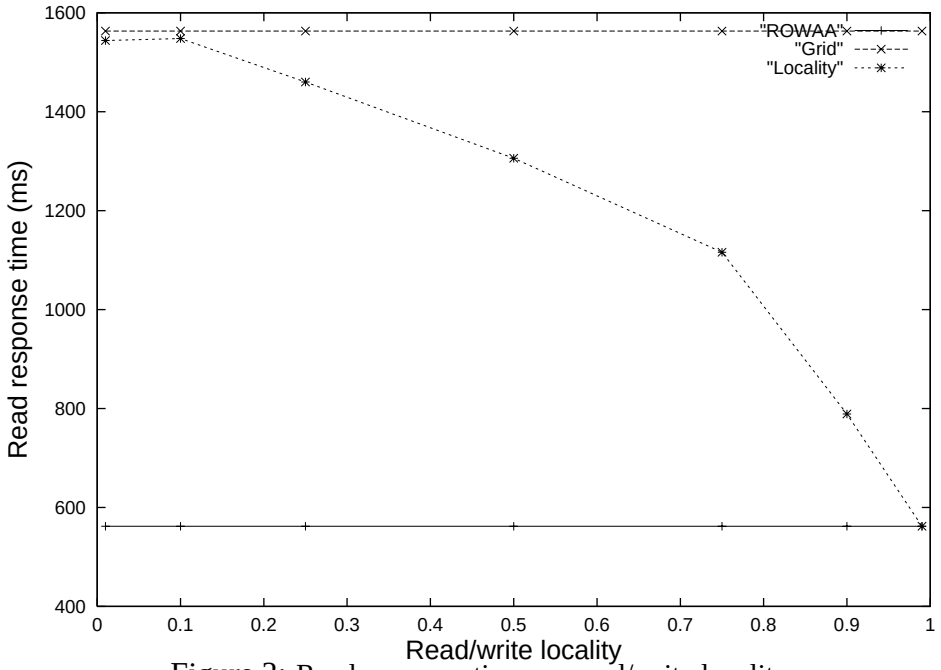

Figure 3: Read response time vs. read/write locality

5.2.2 Read/write ratio

Graph 2 indicates the impact of workload to the overall system scalability. To magnify the result, we choose the replication size to be 100. In this graph, the x-axis represents the write rate in the given workload, and the y-axis represents the overall system load in terms of the combined number of operations performed at all replicas. Notice that the relatively flat curve in the middle represents the grid protocol. The scalability of the grid protocol is insensitive to write rate because its both read and write operations has the same cost of $\sqrt{n}$. In the case of ROWAA protocol, its limited scalability to handle writes becomes obvious as the write rate increases in the given workload. Our locality based protocol at the 90/when write rate increases. The increase in the overall replication cost is due to the occasional invalidation process when some writes from the same client are routed to a replica outside the original write quorum. The probability for an invalidation process to be initiated in the system can be reduced by requiring a stronger locality in the request routing process.

5.2.3 Response time

One important issue to address for the locality based protocol is the response time reads. Since reads dominates the web traffic, most of web applications/services are required to quickly respond to their client inquiries. Graph 3 shows how locality affects the response time of reads. We fix

both the read-to-write ratio to 50/replication size to 100, which result in that the read costs of both ROWAA and grid protocol are of constants. The x-axis represents the read/write locality and the y-axis represent the read response time. The graph confirms that the reads take longer time to process when the write/read locality is low. As long as a client read could not obtain the value locally from the replica it is routed to, additional around trips are necessary for this replica to query from a read quorum to obtain the fresh value. Since most existing server selection algorithms consider the locality based routing policy to increase cache hits on server replicas, the replication system utilizing such routing policy automatically provides a strong read/write locality. Therefore, the locality based protocol provides the read response time that is competitive as that of ROWAA, illustrated by the middle curve in the 90/

6 Conclusion

In this work, we exploit the workload and replication scale impact to scalability and response time of replication systems. Although the popular ROWAA protocol has good scalability and read response time under read-dominated workload, it suffers from the high write cost as the write ratio increases in the workload. The grid quorum protocol has constant write cost across all read-to-write ratio. But its read cost is much higher than that in ROWAA so that the grid quorum approach is undesirable for the majority of web applications, which handles workload with a high fraction of reads.

The locality based protocol is able to achieve the low write cost as in the grid approach and the low read cost as in the ROWAA approach under the assumption that requests from the same client are routed to the same replica most of times. Such a strong read/write locality assumption can is realized in the clustering systems that route requests of the same client to the same server replica to increase cache hits. As the conclusion, the scalability benefit of the locality based replication protocol can be realized in the existing replication systems with minimal modification to the current request routing policy.

References

[1] D. Agrawal and A. Abbadi. The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data. In *Proceedings of the Sixteenth International Conference on Very Large Data Bases*, 1990.

[2] P Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control adn Receivery in Database Systems*. Addison Wesley, 1987.

[3] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–188, June 1994.

[4] S. Cheung, M. Ahamad, and M. Ammar. The grid protocol: a high performance scheme for maintaining replicated data. In *Proceedings of the Sixth International Conference on Data Engineering*, pages 438–445, 1990.

[5] L. Gao, M. Dahlin, A. Nayate, J. Zheng, and A. Iyengar. Application-specific content distribution for edge services. In review., November 2002.

[6] S. Gribble, E. Brewer, J. Hellerstein, and D. Culler. Scalable, Distributed Data Structures for Internet Service Construction. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation*, October 2000.

[7] R. Jimenez-Peris, M. Patino-Martinez, G. Alonso, and B. Kemme. How to Select a Replication Protocol According to Scalability, Availability, and Communication Overhead. In *IEEE Int. Conf. on Reliable Distributed Systems (SRDS'01)*, October 2001.

[8] J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[9] V. Paxson. End-to-end Routing Behavior in the Internet. In *Proceedings of the ACM SIGCOMM '96 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 1996.

[10] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. Demers. Flexible Update Propagation for Weakly Consistent Replication. In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, October 1997.

[11] R. Thomas. A Majority Consensus Approach to Concurrency Control for Multiple Copy Database. In *ACM Transactions on Database Systems*, pages 180–209, June 1979.

[12] H. Yu and A. Vahdat. Minimal Cost Replication for Availability. In *Proceedings of the Twenty-First Symposium on the Principles of Distributed Computing*, 2002.

[13] Y. Zhang, V. Paxson, and S. Shenkar. The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. Technical report, AT&T Center for Internet Research at ICSI, http://www.aciri.org/, May 2000.