Hybrid Replication: Optimizing Network Bandwidth and Primary Storage Performance for Remote Replication

Assaf Natanzon*‡, Philip Shilane*, Mark Abashkin*‡, Leehod Baruch*, and Eitan Bachmat‡

* EMC Corporation
Email: firstname.lastname@emc.com

‡ Department of Computer Science
Ben-Gurion University of the Negev
Email: ebachmat@cs.bgu.ac.il

***Abstract*—Traditionally, there are two main forms of data replication to secondary locations: continuous and snapshot-based. Continuous replication mirrors every I/O to a remote server, which maintains the most up-to-date state, though at a large network bandwidth cost. Snapshot replication periodically transfers modified regions, so it has lower network bandwidth requirements since repeatedly overwritten regions are only transferred once. Snapshot replication, though, comes with larger I/O loads on primary storage since it must read modified regions to transfer. To achieve the benefits of both approaches, we present hybrid replication, which is a novel mix of continuous replication and snapshot-based replication. Hybrid replication selects data regions with high overwrite characteristics to be protected by snapshots, while data regions with fewer overwrites are protected by continuous replication. In experiments with real-world storage traces, hybrid replication reduces network bandwidth up to 40% relative to continuous replication and I/O requirements on primary storage up to 90% relative to snapshot replication.**

Keywords: replication, data protection, continuous data protection, snapshot

I. Introduction

Remote replication is a widely used technique to protect primary storage systems by generating the same state on a secondary system [5], [10], [14], [16], [18]. Unlike the data protection options of RAID and creating explicit copies, remote replication creates a copy on a separate machine that may be geographically near or distant. Depending on the style of remote replication, the secondary system may be part of a high-availability solution that can handle primary storage tasks in the event the main system is offline. Our work focuses on improving the efficiency of remote replication (simply called replication in the remainder of this paper) for backups and high availability solutions.

Replication techniques have been studied extensively [5], [10], [14], [16], [18], and we focus on the two most commonly implemented approaches to protect storage volumes: continuous [5] and snapshot [10] replication. Continuous replication is a conceptually simple technique that intercepts every write I/O to a protected volume and sends writes to both the local storage volume as well as to the replica site. Continuous replication has two main advantages over snapshot replication. First, it supports a very low Recovery Point Objective (RPO) [17], meaning a replica only lags the primary system by a short period, perhaps only a few seconds. Second, since data is sent to the replica inline, it is unnecessary to read data from the primary system, unlike snapshot replication. The main disadvantage of continuous replication is that the network bandwidth requirements can be quite high, and, specifically, as high as peak client writes to the primary system.

In snapshot-based replication, the storage system periodically creates snapshots of a volume and tracks changes between snapshots. When new writes enter the system, the snapshot content is protected so both the earlier and current content are available. Then, when replication starts, it reads the modified regions from disk and transfers the regions across the network to the replica. Because there may be numerous overwrites to the same region, and only the final version within a snapshot period needs to be transferred, the amount of bandwidth used can be dramatically lower than for continuous replication. Previous work shows bandwidth may be 80% lower with hourly snapshots [10], [14]. Snapshot replication generally has longer RPOs than continuous replication, extra storage space to track changes, and I/O to read modified data from primary storage.

The goal of hybrid replication is to achieve the network bandwidth efficiencies of snapshot replication and low overheads on primary storage of continuous replication. The basic approach is to partition a volume into *extents*, consisting of consecutive blocks, that are protected by either continuous or snapshot replication. Hybrid replication assigns extents with the most overwrites to snapshot replication, and the remaining extents are assigned to continuous replication. As storage patterns change, extents are reassigned dynamically. Hybrid replication will have the RPO of snapshot replication but reduces primary I/O overheads up to 90% versus snapshot replication while also reducing network bandwidth overheads up to 40% versus continuous replication.

The remainder of our paper is organized as follows. Section II presents background on continuous and snapshot replication, followed by related work in Section III. The hybrid replication algorithm is described in Section IV. Our methodology for evaluating hybrid replication is described in Section V. Experiments are presented in Section VI, and the last section presents our conclusions and future work.

II. Remote Replication Background

We next present more detailed background information about continuous and snapshot replication that motivates our work on creating a hybrid solution. We use Figure 1 as an ongoing example to motivate hybrid replication.

978-1-5090-3315-7/16/$31.00 ©2016 IEEE

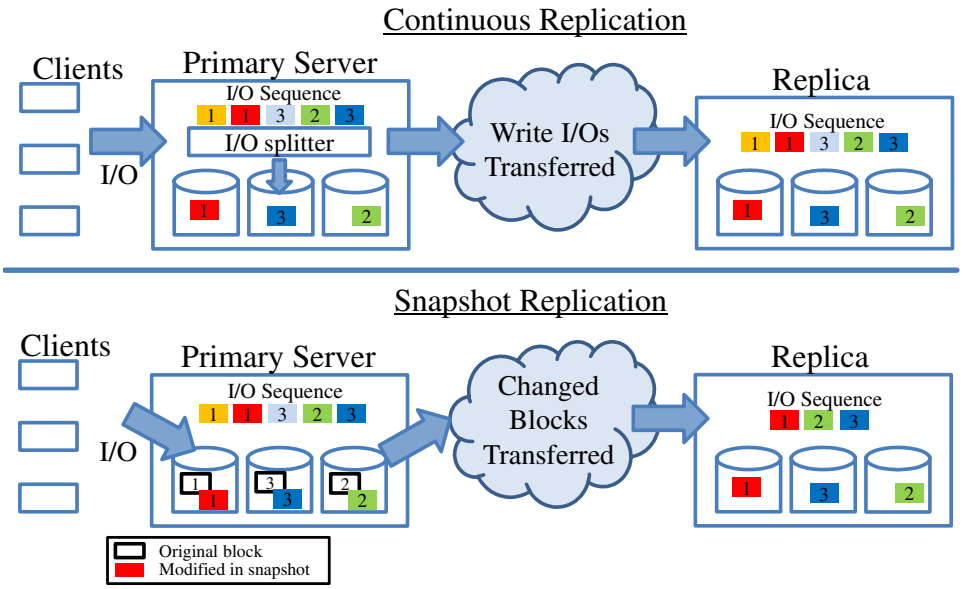


Fig. 1. An example of continuous and snapshot replication. Continuous replication tends to have higher network bandwidth requirements, while snapshot replication tends to create more I/Os on the primary server.

A. *Continuous Replication*

An example of continuous replication is shown in the top half of Figure 1. Starting at the left, client I/Os are sent to the primary storage system. While the primary system serves both reads and writes, we will focus on write traffic as it is the modified data that must be replicated. In this example, write I/Os are directed to logical block addresses (LBAs) 1, 1, 3, 2, and 3, with the repeated writes to 1 and 3 creating overwrites. Each write is assigned a unique color to highlight differences between techniques. In continuous replication, all of the I/Os are intercepted by an I/O splitter in the primary storage system that directs the writes to the primary system as well as to the replica, which receives the same I/O write pattern. Note that write I/Os can be sent to the replica without ever reading regions back from primary storage. This example shows that repeated writes to LBAs 1 and 3 are transferred to the replica, which increases network bandwidth usage, though it does enable a shorter RPO. The final region state is shown on disk after the sequence of I/Os.

Continuous replication techniques include synchronous replication [8], which send all of the write I/Os to the replica, as we described. Synchronous replication may be configured to wait for confirmation from the replica before acknowledging the write was received from the client, creating an RPO of zero, though client acknowledgments have a delay due to network transfer. An alternative version sends the data to the replica but responds to the client immediately without waiting for a confirmation from the replica, which has a faster response to the client but the RPO then includes the network transfer time. When a client requires the lowest RPOs, continuous replication is preferred over snapshot-based approaches, including hybrid replication.

There are also asynchronous versions of continuous replication where the data is buffered in memory and sent asynchronously. Data buffering supports limited write-coalescing for overwrites that take place within a short period of time. In this example, repeated writes to LBA 1 could have resulted in a single I/O sent to the replica with buffering. Buffering helps control bandwidth fluctuations, and even a relatively small buffer may reduce the maximum required bandwidth of the system though the RPO increases with buffering delays.

One example of a commercially available continuous replication system is EMC's RecoverPoint, which installs an I/O splitter on the primary storage system such as a storage array or hypervisor and supports the spectrum of synchronous and asynchronous replication [9]. The overhead of adding a splitter may be as low as 150 microseconds added to writes, which is low relative to creating and maintaining snapshots.

B. *Snapshot Replication*

Snapshot replication is shown in the bottom half of Figure 1. The main difference between snapshot and continuous replication is that client writes land on the primary storage system and are later read and transferred to the replica. Because only the most recent version of a region written within a snapshot period is transferred, network bandwidth is lower for snapshot replication as compared to continuous replication. In this example, LBAs 1, 2, and 3 are transferred once at the end of the snapshot period instead of transferred once per overwrite from the clients. As the snapshot period increases, network usage typically decreases until a plateau is reached. As part of the snapshot process, though, extra copies are created on the primary storage system for modified regions. This is shown in the figure with the original regions outlined and the modified versions colored. When replication has completed, the extra space for a snapshot can be released once the modified regions are merged into the active region set.

Various techniques have been explored to reduce storage overheads during snapshot replication. Using change trackers allows the system to erase a snapshot once it has been

replicated, so it is unnecessary to wait for the next snapshot to be created [14], but the current snapshot state still must be preserved until it is replicated. There are two basic snapshotting mechanisms used to preserve the original content until replication completes: copy-on-write and redirect-on-write. With copy-on-write, when an incoming write to LBA 1 is received, the original content is copied to a new location, and the incoming write is then applied to LBA 1. This adds significant cost to every write and decreases storage performance [2], [3]. As an alternative, redirect-on-write preserves content in its current location and redirects incoming writes to new locations. VMware uses a variant of redirect-on-write that uses a log device when creating snapshots. When the snapshot is released, the log data is copied to the main volume, which is a storage bandwidth overhead [6]. As a further variant, NetApp WAFL FS uses a log-structured file system, so that new writes are continuously appended to the log, which requires a garbage collection mechanism [4].

Snapshot based replication has been implemented both for storage arrays such as NetApp's Snap Mirror [10] as well as in virtualization environments. For example VMware supports vStorage APIs for Data Protection [15], which supports the creation of snapshots of virtual machines (VMs), change block tracking, and reading the changed blocks from the snapshot in order to create an efficient backup of the VMs. Even flash-based storage can suffer from I/O overheads of snapshot replication. We have spoken with customers where snapshot replication decreased overall IOPS by up to 25% on their flash-based system. In practice many customers face replication performance issues, so they often schedule snapshot replication outside of work hours.

III. Related Work

Replication has been implemented in numerous styles, and we briefly summarize several overview articles. Ji et al. [5] created a thorough taxonomy of replication techniques based on numerous properties including the fault coverage model, synchronization threshold, how updates are propagated, when updates are acknowledged, and where replication takes place. Our hybrid replication technique falls between categories outlined in their work as it is a combination of **in-order asynchronous** and **write-coalescing batches with atomic update** using their terminology. Mirzoev [8] provides a description of synchronous and asynchronous replication for block storage. Azagury et al. [2] discuss multiple techniques to create point-in-time copies.

Continuous replication is also sometimes called synchronous remote mirroring. Since every I/O is mirrored, and in some implementations I/Os block until the replica acknowledges the I/O, it is more commonly implemented when the primary and replica are geographically near each other. Natanzon et al. [9] discuss synchronous and asynchronous options. Weatherspoon et al. [16] presented an alternative approach they call network-sync, which uses error-correcting packets to reduce the network latencies of traditional synchronous replication as long as the rate of packet loss is low in the network. Zhang et al. [18] added remote replication to the iSCSI protocol and demonstrated consistency even with network delays of up to one second. They also noted the benefits of write-coalescing to reduce bandwidth requirements.

Snapshot creation is a common feature in multiple storage systems including ZFS [3], and Btrfs [12], though replication is not always natively supported. Snapshot replication can introduce storage overheads on the primary server, as both the original regions and modified regions must be maintained until replication completes and the original regions can be freed. Multiple improvements have been studied to address this drawback. Patterson et al. [10] describe a technique for tracking and replicating regions that changed between snapshots. Shim et al. [14] introduced replication snapshots, which use either redirect or copy-on-write only for regions which were overwritten since the last snapshot was replicated, otherwise the write can be applied in-place without affecting consistency. For transactional applications, there has been work on creating consistent snapshots at scale [1], [11], [13]. While most of these techniques reduce storage overheads for snapshots and support consistency, hybrid replication addresses the issue that data must still be read from storage to be replicated.

A related technique to replication is live migration, which transfers a live VM to improve load balancing or to continue running a VM during hardware upgrades. Mashtizadeh et al. [7] described VMware's live migration algorithm, which leverages a combination of snapshotting and I/O mirroring. Unlike live migration, which transfers a VM to a second location in a best-effort time frame, hybrid replication provides ongoing data protection with a RPO.

IV. Hybrid Replication Architecture

A. Overview

The two main goals of hybrid replication are to reduce both network bandwidth and primary storage I/O. Our algorithm achieves these goals by applying snapshot replication to extents that are frequently overwritten (reducing network bandwidth requirements) and continuous replication to extents that are rarely overwritten (reducing primary storage I/O). For each time period corresponding to a client's configured RPO, the volume is partitioned into snapshot and continuous zones, where zones are a logical view managed by an I/O splitter and replication software, while the physical storage layout is unaffected. Incoming writes to the continuous zone are split and applied both locally and transferred to the replica, while blocks modified in the snapshot zone are read from storage and transferred at the end of the time period.

Consider an example of hybrid replication shown in Figure 2. Frequently written data extents are colored in red, rarely written data extents are colored in blue, and extents with a medium number of writes are colored yellow. This is a simplified example, and actual overwrite counts will fall on a spectrum from a high value to zero per extent. Given a configurable threshold for the number of extents that can be protected by a snapshot, we assign extents with the highest overwrite rate to the snapshot zone, and the remaining extents are protected with continuous replication. In this figure, snapshot-protected extents are marked with a dashed outline

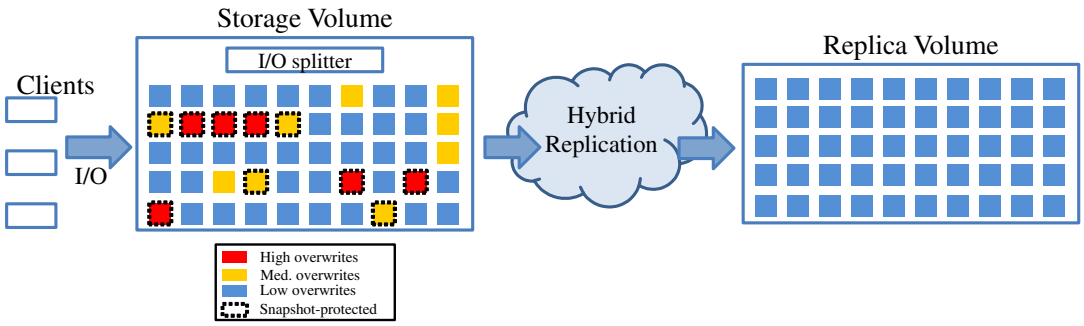


Fig. 2. The storage volume shows extents marked as receiving high, medium, or low overwrites. 20% of the extents with the largest number of overwrites are selected for snapshot-based replication, while the remaining extents (including several with medium overwrites) are protected with continuous replication.

for 20% snapshot protection and 80% continuous protection. When a write enters the server, the I/O splitter checks whether it is directed to an extent in the snapshot or continuous zones, and either applies copy-on-write (or redirect-on-write) or immediate remote transfer, respectively. At the end of the period, modified extents in the snapshot zone are read and transferred to the replica. This generates a consistent image of the volume at the replica.

The hybrid replication algorithm is defined by two parameters: the fraction of the system protected by continuous replication and the RPO. We refer to hybrid replication as $X\%$-continuous if data in $X\%$ of the volume is in the continuous zone and the rest of the volume is protected by snapshot replication. The RPO is the maximum amount of time in which data can be lost in case of a primary storage disaster, and it is equal to twice the period for taking a snapshot. This is because, until we finish shipping a snapshot created at time t, in the event of a storage failure, we will have to return to the previous snapshot created at time $t-1$. As an example, 0%-continuous replication with an RPO of 30 minutes is purely snapshot-based replication with snapshots created every 15 minutes. The opposite extreme is 100%-continuous, which means that every I/O is mirrored to the replica, and the RPO parameter is relevant to the buffered form of continuous replication but is zero for the synchronous version.

Our hybrid approach offers multiple advantages. First, it allows us to make trade-offs between the amount of bandwidth required and the number of disk I/Os on a primary volume depending on customer costs. Second, hybrid replication tends to smooth network bandwidth requirements relative to snapshot-only replication, which waits until a snapshot period ends to begin transferring all of the changes. Third, hybrid replication supports dynamic network throttling by reassigning extents from continuous to snapshot protection. In order to achieve these advantages, though, hybrid replication has a larger RPO than strictly continuous replication.

In the following subsections, we provide more details about assigning extents to continuous and snapshot zones. We then discuss implementation details related to network bandwidth management and handling peak workloads.

B. Selection of Hybrid Zones

Assigning portions of the volume to either continuous or snapshot protection is the key step in hybrid replication. While a trivial assignment of 50% of the volume to continuous and 50% to snapshot-based would reduce network bandwidth and primary I/O, a more targeted assignment algorithm can produce greater benefits. We first discuss an optimal algorithm before discussing our prediction approach.

Locations which have significant overwrites should be assigned to the snapshot zone, while locations which have fewer overwrites should be assigned for continuous replication. In this way, hybrid replication could take advantage of overwrites to reduce bandwidth. First consider an optimal division of a volume with a RPO of $2t$ minutes that creates a snapshot every t minutes. An optimal algorithm would assign small extents of the volume to the appropriate zone, where the extents could be as small as 4KB since that is often the minimal I/O write size for many block devices. For each 4KB extent, we consider the number of overwrites that take place within the t minutes covered by the snapshot. We then sort the extents by the number of overwrites each extent receives. An optimal hybrid algorithm that is $X\%$-continuous will assign $X\%$ of the extents with the lowest overwrites to the continuous zone. The remaining $100-X\%$ of extents are assigned to the snapshot zone. This algorithm is optimal because the extents in the snapshot zone are exactly the extents which are overwritten the most frequently.

Since the optimal algorithm requires future knowledge of write patterns, we create an approximation algorithm that attempts to predict the number of overwrites per extent during a snapshot period. First, our algorithm divides the volume into extents, which span multiple consecutive 4KB blocks to reduce memory overheads, since tracking information is needed for each extent. For each extent we track the number of bytes written to the extent during previous snapshot periods. Then, in each period we calculate the amount of data written to the extent divided by the size of the extent. We use the average amount of data written to an extent per hour in a rolling window, though functions weighted based on recency could be considered in the future. The higher the value for an extent, the more likely the extent will be written repeatedly in the next period.

We now assign extents to either continuous or snapshot zones. Like in the optimal algorithm, we assign the $X\%$ of extents with the fewest predicted overwrites to the continuous zone, and we assign the remaining $100-X\%$ of the extents (the most overwritten) to the snapshot zone. This applies to

extents that received writes in previous periods, but a subtle point is how to assign extents that received zero writes in previous periods. We randomly assign $X\%$ of such extents to the continuous zone and the remainder to the snapshot zone to preserve the desired $X\%$-continuous protection, on average. Since we expect few writes to these extents, their assignment often has little impact on overall results. While our prediction algorithm is fairly simple, in practice we have found that it achieves large reductions in network bandwidth and primary storage I/O. The prediction algorithm could be further improved in future work within the general framework of hybrid replication.

To provide further implementation details, the above algorithm can be split into two main phases. The first phase tracks overwrites to each extent, and the second phase makes a prediction about future overwrites to assign extents to zones. The I/O splitter is a natural place to add overwrite tracking, since that is early in the storage pipeline, and the splitter can communicate with the replication software. Then the replication software can assign extents to zones. Also, while we generally use large extents (e.g. 256KB or larger) for overwrite tracking and zone assignment, the snapshot system maintains finer tracking for change blocks. Since overwrite tracking is maintained in memory it must be compact, but for change tracking, we append meta data records to a disk log. By scanning the log, we can identify and transfer the modified sectors (512B), which reduces the amount of data read from disk and transferred across the network relative to only tracking extents as dirty or clean. When the snapshot period (i.e. 20 minutes) ends, we begin to transfer modified data from the snapshot zone, reassign extents to zones for the next period, and begin tracking new writes.

As I/O patterns change over time, the assignment of extents to either the continuous or snapshot zones will change between each snapshot period. While our current implementation takes %-continuous as a fixed parameter, in future work we plan to explore modifying this value between snapshot periods based on network bandwidth, network quality, the priority of storage volumes, and other properties.

C. Bandwidth Management

Given a fixed network bandwidth between the primary server and the replica, our technique must allocate bandwidth between the continuous replication and snapshot replication components of hybrid replication. During replication, both continuous and snapshot replication utilize memory buffers during network transfer. Continuous replication fills its buffer from the I/O splitter, while snapshot replication fills a buffer by reading modified data from primary storage.

The hybrid algorithm's first priority is to send the continuous data, and when the continuous buffer drops below a fullness threshold (e.g. 10MB), snapshot data can be transferred. The fullness threshold determines the amount of space needed to handle bursty writes until new writes can be transferred. We calculate how much bandwidth is needed to transfer snapshot data within t minutes (i.e. the snapshot period) and attempt to give each replication technique proportional bandwidth. During peak client writes, continuous bandwidth will receive all of the bandwidth, but during lulls in client writes, snapshot replication can take more network bandwidth.

D. Peak Workload Management

While hybrid replication reduces the amount of bandwidth required relative to a purely continuous replication technique, bursty writes from clients may exceed network bandwidth availability. One option is to use large memory buffers to hold the bursty writes, but in some extreme situations continuous replication may exhaust available memory buffers. In this situation, some continuous replication algorithms delay client I/Os as a throttling mechanism. Other implementations may pause continuous replication and restart network transfer once the peak write period ends.

Our hybrid replication algorithm takes the approach of dynamically reassigning extents from continuous to snapshot zones during peak periods. When the continuous replication buffer reaches its capacity, the extent with the largest number of overwrites in the buffer is reassigned to the snapshot zone. Among extents in the buffer, this decision creates the greatest reduction in network bandwidth due to write-coalescing, though it is not optimal in the broader sense since it only considers extents in the current buffer.

V. Methodology

To evaluate hybrid replication, we analyze real storage traces from customer environments using a detailed simulator.

A. Storage Traces

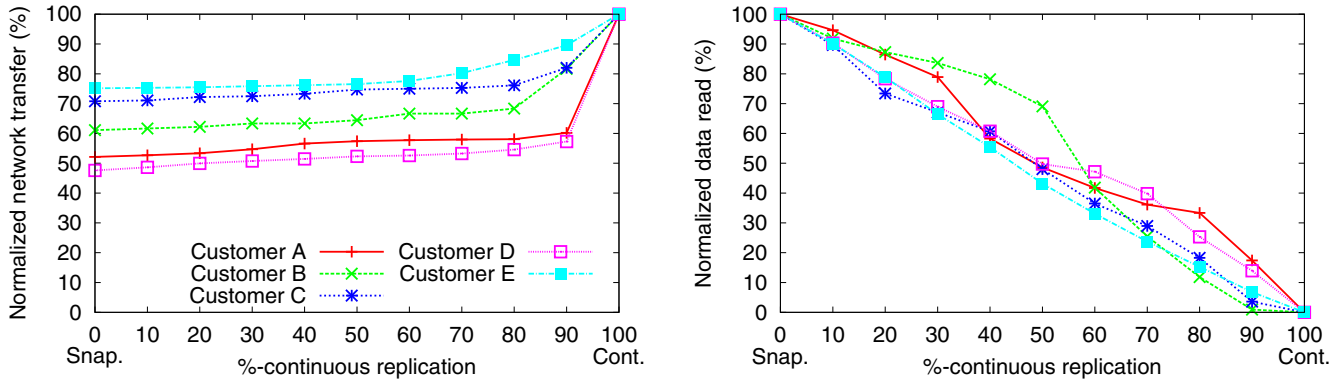
We analyze traces from five EMC Symmetrix production systems collected at customer sites, comprising a total of around 7700 volumes [14]. The machines vary from the smallest machine with 34 volumes to the largest with almost 1500 volumes. Volume sizes vary from a few gigabytes to multiple terabytes. The collection period lasted at least 24 hours and recorded millions of I/Os per hour. Unfortunately, we do not have a mapping from applications to volumes due to limitations of the collection process, but applications include all types of enterprise applications including SAP, Oracle, and Exchange mail servers. We refer to the traces as `Customers A-E` in Section VI.

The traces record each I/O to the system including the following information:

- **time stamp**: indicating when the request was received
- **I/O operation**: read or write
- **logical unit ID**: the volume targeted by the I/O request
- **logical block address**: the offset of the start of the request within the volume, in units of 512 byte sectors
- **size**: I/O size in sectors

B. Simulation

Our analysis is performed with a trace-driven simulator that parses storage traces and implements hybrid replication while generating statistics about network bandwidth requirements,



(a) Network transfer

(b) Primary data read

Fig. 3. Normalized results across all customers with an extent size of 4MB and snapshot period of 5 minutes. Hybrid replication transfers less data across the network than continuous replication and reads less data from primary storage than snapshot replication.

primary storage I/O, and other results. Configuration parameters include snapshot period, %-continuous, extent size, buffer sizes, etc. We experimented with extent sizes of 256KB, 1MB, 4MB, 16MB, and 64MB as well as snapshot periods of 5, 10, 15, and 30 minutes, though our simulator supports other values. We generally found few differences for reasonable buffer sizes, so we neglect those experiments from our evaluation. From experience with commercial replication solutions, we believe that these simulation results approximate what can be achieved with a full implementation.

VI. Evaluation

We first present overall results for all five customers in terms of data transferred across the network and data read from primary storage, then focus on two customers to demonstrate the impact of varying the snapshot period, and finally analyze the interaction of extent size and memory overheads.

A. Overall Results

We begin by showing results across all five customers in terms of data transferred across the network and primary data read in Figure 3. These experiments were configured with a 5 minute snapshot period, as taking a snapshot every 5 minutes is on the more aggressive end for snapshot approaches. We also use 4MB extents since it provides a trade-off between reducing memory overheads without overly coarse tracking granularity, as discussed in Section VI-C.

The horizontal axis shows %-continuous varying from 0% (snapshot replication) up to 100% (continuous replication). Since the amount of data involved varies by hundreds of gigabytes between customers, we have normalized the vertical axes by the largest amount of data either transferred or read per customer. As a comparison, the leftmost data points represent the results for snapshot replication, while the rightmost data points show the results for continuous replication, with hybrid replication shown as a percentage change between the endpoints. Figure 3(a) shows that bytes transferred across the network tend to grow slowly until somewhere in the 80-90%-continuous range and then increases rapidly. Figure 3(b) shows

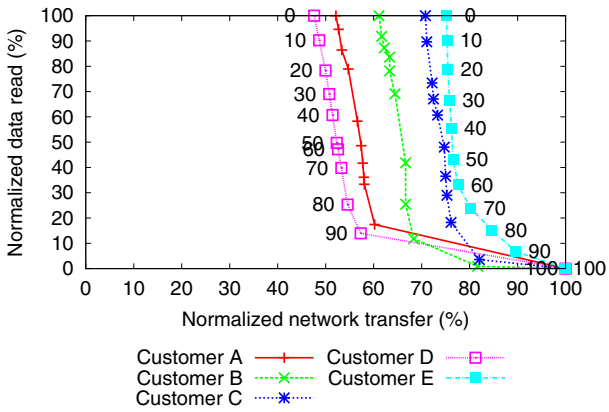
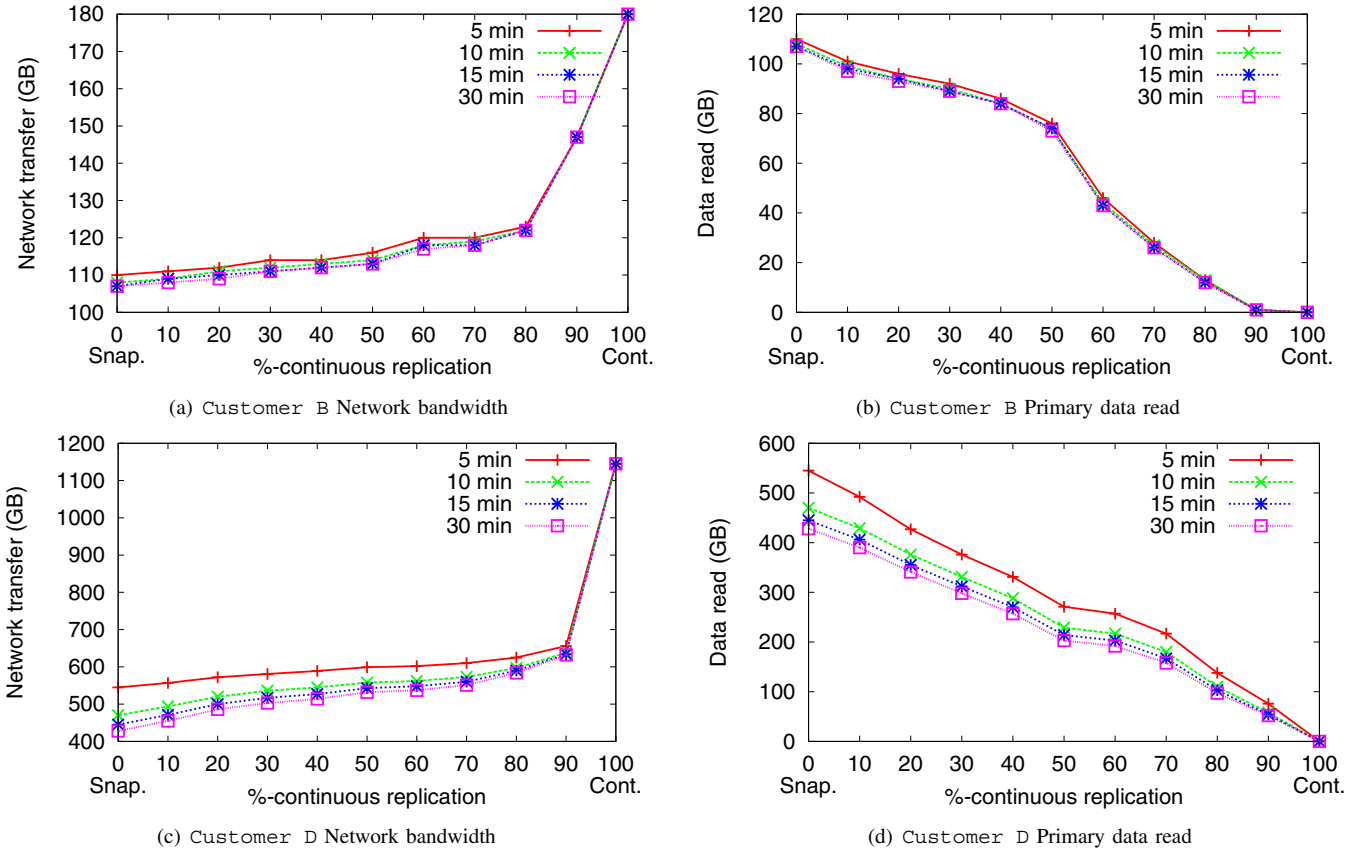


Fig. 4. Optimizing %-continuous for both network transfer and data read suggests a value of 80-90% is best for all customers. Better results are towards the bottom left of the figure.

that data read from primary storage decreases fairly linearly for most customers as %-continuous increases.

We next study optimizing the combination of network transfer and data read. Figure 4 shows the normalized network transfer on the horizontal axis and the normalized data read on the vertical axis. Each customer's result is plotted, with labels for 0%-100%-continuous on the leftmost and rightmost customers in the figure for clarity reasons. Results towards the bottom left of the figure represent reduced system overhead as compared to results strictly above or to the right. For `Customers A-D`, the results drop rapidly as %-continuous increases until a knee in the curve around 80-90%. `Customer E` has a more gradual curve because of fewer overwrites, so at 80%-continuous, there is a 80% savings in primary data read and a 15% savings in network transfer.

To optimize %-continuous automatically, a system must consider the RPO specified by a customer as well as the cost of network bandwidth and primary I/O. Some customers have excess bandwidth, while others have excess primary I/O. If the optimal %-continuous value is selected for these customers based on an equal weighting for overheads, values of either 80% or 90% would be selected. This would reduce network



(a) Customer B Network bandwidth

(b) Customer B Primary data read

(c) Customer D Network bandwidth

(d) Customer D Primary data read

Fig. 5. The impact of snapshot period and %-continuous on both network transfer and data read. Note that the vertical axis has different scales to reflect the differences in customer data sizes and to highlight the impact of snapshot period.

traffic by 15-40% for continuous replication and primary data read by 80-90% for snapshot replication. These results confirm that hybrid replication is a promising replication technique to reduce data protection overheads.

B. Impact of Snapshot Period

We next focus on Customers B and D to highlight differences due to snapshot period, as shown in Figure 5. Starting with Customer B, on the far left, we see that the amount of data transferred is highest for a snapshot created every 5 minutes and lowest for a snapshot created every 30 minutes because more writes are coalesced during the longer period. As the %-continuous increases, network bandwidth grows slowly until 80%-continuous when it grows rapidly. Note that 100%-continuous has the same network requirements regardless of the snapshot period. The amount of data read from primary storage is shown in Figure 5(b) with the amount of data read decreasing as hybrid replication approaches 100% continuous replication. These results demonstrate that configuring hybrid replication at 80%-continuous for this customer achieves most of the network transfer savings of snapshot replication and the primary data read savings of continuous replication across snapshot period. Because of few overwrites beyond the snapshot periods, there are few differences between snapshot periods for Customer B.

The results for Customer D shown in Figures 5(c) and 5(d) show similar trends. Network transfer increases and primary data read decreases with %-continuous. A difference between Customers B and D is that the impact of snapshot period is larger for Customer D as shown by a larger spread between snapshot period results. This demonstrates the impact of more frequent overwrites beyond 5 minutes for Customer D than B though hybrid replication remains effective.

C. Extent Size

Finally, we investigate the impact of extent size on hybrid replication. In this experiment, we configured each client with a snapshot period of 5 minutes at its optimal %-continuous value, either 80% or 90%. In terms of network transfer, we found that extent size had little impact since customer writes to the continuous zone are transferred immediately and are unaffected by extent size. On the other-hand, Figure 6 shows that extent size affected three of the five customer traces. As the extent size increases along the horizontal axis, the normalized data read from primary storage decreases for Customers B, C, and E and is stable for the other two customers. These three customers had more sequential access patterns, and larger extents better represented zone assignments, so that nearby overwrites were assigned to the snapshot zone.

Memory overheads are plotted using the right vertical axis assuming a 100TB system and 40 bytes per extent record. With

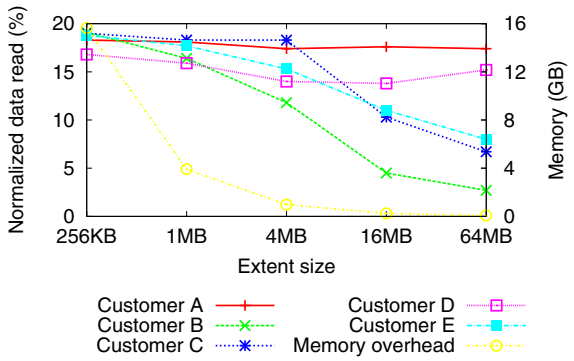


Fig. 6. As the extent size increases, the amount of data read is either stable or decreases. Note the left vertical axis is zoomed to highlight the data read results, and the right vertical axis shows memory overheads.

4MB extents, the memory overhead is under 1GB, which is reasonable for this capacity. This result suggests that large extent sizes can be used to both decrease memory overheads as well as the amount of data read from primary storage with little impact on data transferred across the network.

VII. Conclusions and Future Work

Replication is a valuable component of data protection and high availability solutions that creates a replica image at a secondary location. For most customers, the choice has been between continuous replication and snapshot replication, which have typically forced customers to either over-provision their network bandwidth or I/O capabilities of their primary server, respectively. We present hybrid replication, a novel technique that assigns extents with numerous overwrites to a snapshot replication zone and extents with few overwrites to a continuous replication zone to reduce such overheads.

In experiments with real customer traces, we have shown that hybrid replication can decrease network bandwidth traffic by up to 40% and data read from primary storage by up to 90%. These reductions enable customers to either decrease costs associated with the network and primary storage I/O, or equivalently, decrease their RPO by creating more frequent replicas. Hybrid replication has several additional benefits. 1. Network traffic is smoothed relative to snapshot-based replication, which begins network transfer at the end of each period. 2. Network bandwidth can be throttled by transitioning extents from the continuous zone to snapshot zone during peak periods. 3. Zone assignment naturally adjusts to changing I/O patterns from customers. In conclusion, hybrid replication is a promising new replication technique that achieves the resource savings of both continuous and snapshot replication while dynamically adjusting to a changing storage environment.

For future work, we plan to investigate further improvements to hybrid replication. While zone assignment currently changes between periods, our current implementation uses a fixed %-continuous value, which should be modified in response to client I/O and network bandwidth changes. Also, the current zone assignment algorithm, while effective, is fairly simple, and advanced prediction techniques may improve accuracy. Finally, we hope to further validate hybrid replication with a more complete prototype.

References

[1] Ardekani, M. S., Sutra, P., and Shapiro, M. Non-monotonic snapshot isolation: scalable and strong consistency for geo-replicated transactional systems. In *Proceedings of the IEEE International Symposium on Reliable Distributed Systems (SRDS)* (2013).

[2] Azagury, A., Factor, M. E., Satran, J., and Micka, W. Point-in-time copy: Yesterday, today and tomorrow. In *Proceedings of the IEEE/NASA Conference on Massive Storage Systems and Technology* (2002).

[3] Bonwick, J., Ahrens, M., Henson, V., Maybee, M., and Shellenbaum, M. The zettabyte file system. In *Proceedings of the USENIX Conference on File and Storage Technologies* (2003).

[4] Hitz, D., Lau, J., and Malcolm, M. File system design for an nfs file server appliance. In *Proceedings of the USENIX Winter Technical Conference* (1994).

[5] Ji, M., Veitch, A., and Wilkes, J. Seneca: remote mirroring done write. In *Proceedings of the USENIX Annual Technical Conference* (2003).

[6] Laverick, M. VMware backup and the VMware snapshot. http://www.computerweekly.com/feature/VMware-backup-and-the-VMware-snapshot, 2012.

[7] Mashtizadeh, A., Celebi, E., Garfinkel, T., Cai, M., et al. The design and evolution of live storage migration in VMware ESX. In *Proceedings of the USENIX Annual Technical Conference* (2011).

[8] Mirzoev, T. Synchronous replication of remote storage. *Journal of Communication and Computer 6*, 3 (March 2009), 34–39.

[9] Natanzon, A., and Bachmat, E. Dynamic synchronous/asynchronous replication. *ACM Transactions on Storage (TOS) 9*, 3 (2013).

[10] Patterson, H., Manley, S., Federwisch, M., Hitz, D., Kleiman, S., and Owara, S. SnapMirror: file system based asynchronous mirroring for disaster recovery. In *Proceedings of the USENIX Conference on File and Storage Technology* (2002).

[11] Peluso, S., Romano, P., and Quaglia, F. Score: A scalable one-copy serializable partial replication protocol. In *Proceedings of the Middleware Conference* (2012).

[12] Rodeh, O., Bacik, J., and Mason, C. Btrfs: The Linux B-tree filesystem. Tech. rep., IBM Research Report RJ10501 (ALM1207-004), 2012.

[13] Sciascia, D., Pedone, F., and Junqueira, F. Scalable deferred update replication. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2012).

[14] Shim, H., Shilane, P., and Hsu, W. Characterization of incremental data changes for efficient data protection. In *Proceedings of the USENIX Annual Technical Conference* (2013).

[15] VMware. VMware vSphere Storage APIs – Data Protection. http://kb.vmware.com/kb/1021175, 2015.

[16] Weatherspoon, H., Ganesh, L., Marian, T., Balakrishnan, M., and Birman, K. Smoke and mirrors: reflecting files at a geographically remote location without loss of performance. In *Proceedings of the USENIX Conference on File and Storage Technology* (2009).

[17] Yang, Q., Xiao, W., and Ren, J. Trap-array: A disk array architecture providing timely recovery to any point-in-time. In *ACM SIGARCH Computer Architecture News* (2006), vol. 34, IEEE Computer Society, pp. 289–301.

[18] Zhang, M., Liu, Y., and Yang, Q. Cost-effective remote mirroring using the iSCSI protocol. In *Proceedings of the IEEE Conference on Massive Storage Systems and Technology* (2004).