A

Dynamic synchronous/asynchronous replication

ASSAF NATANZON,, EMC, Ben-Gurion University
EITAN BACHMAT, Ben-Gurion University

Categories and Subject Descriptors: B.4.3 [**Interconnections**]: Synchronous/asynchronous operation

General Terms: Management

Additional Key Words and Phrases: Remote replication

ACM Reference Format:
ACM Trans. Storage V, N, Article A (January YYYY), 17 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Synchronous replication is highly desired in todays IT organizations as any data loss may have significant monetary effect on the organization. In *synchronous replication* every write is acknowledged before more data is replicated. Due to the speed of light limitation on information transfer and the typical two round trip SCSI write protocol, synchronous replication is usually limited to a 50 mile distance. Beyond such a distance the performance slowdown incurred by synchronous writing becomes prohibitive. Even customers who can afford maintaining a replica within a 50 mile distance, may have significant performance issues with synchronous replication. Synchronous replication is not only limited by the speed of light, but is also limited by the WAN bandwidth between the sites and the available storage bandwidth at the replica site. For example, if the WAN bandwidth is lower than the production bandwidth, performance of volumes replicated synchronously, will be limited to the bandwidth available by the WAN. The same is true for the replica storage bandwidth. If the current throughput available for the replica storage is lower than the available production throughput, synchronously replicated applications will slow down to the performance of the replica storage which may have different hardware. For some applications synchronous replication is desired, but low latency is significantly more important. In some cases there are temporal fluctuations in the importance of data loss. For example, many enterprises run batch operations and data warehouse operations, after working hours, or during weekends. The batch operations usually consume significantly more bandwidth than standard daily workloads, but produce less critical data, i.e., data lost during a batch operation can be easily reproduced.

As opposed to synchronous data replication, *asynchronous replication* allows the system to continue replicating data remotely even if not all previous writes have been

Author's addresses: A. Natanzon EMC Corp., Herzlia, Israel and department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel, 84105. assaf.natanzon@emc.com; Eitan Bachmat, Department of Computer science, Ben-Gurion University, Beer-Sheva, Israel, 84105. ebachmat@cs.bgu.ac.il.
The second author was partially supported by an ISF research grant 580/11, And an EMC research grant.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

acknowledged, at least up to a certain lag. This mode of replication is much more performance efficient, but exposes the user to data loss in proportion with the acknowledgment lag, [Keeton et al. 2004]. In cases where the performance penalty of synchronous replication is too prohibitive and disruptive for normal operations, the user had to choose asynchronous replication, with the associated data loss risks.

In this paper we present a method for dynamically switching between synchronous and asynchronous replication. Our method was implemented as part of the Recover-Point network based heterogeneous replication product, which allows continuous replication. The dynamic synchronous/asynchronous replication allows users to choose the maximum allowed write latency per I/O. If latency grows beyond the allowed write latency, the system will automatically move to *semi-synchronous* replication, acknowledging I/O at the production site, before sending the I/O to the replica site. Semi-synchronous replication allows the system to work through short periods of peak loads, but if there is significant load for a longer period, system resources (such as the memory buffer containing the semi-synchronous uncommitted data) may become over utilized. If the memory buffers of the system become full, the system will fall back to an extreme mode of asynchronous replication, *snapshot shipping based replication*. In this mode, the system sends snapshots, i.e., time frozen images, from the production site to the replica site periodically. As resources become available again the system will move automatically from snapshot shipping technology, to semi-synchronous replication and then back to synchronous replication. Recover-Point dynamic replication is generally available and is installed at thousands of customer sites.

The dynamic synchronous/asynchronous replication solves the need for an IT expert to disable synchronous replication when large batch operations are handled, and helps storage system administrators handle slow application response times when synchronous replication is used. System administrator would like to minimize data loss but in many cases cannot compromise production response times. The method we suggest allows the system to provide synchronous replication and no risk of data loss most of the time and allow minimal data loss only on extreme cases (depending on the maximum allowed latency). While the method is opportunistic, for most use cases it is significantly better than plain asynchronous replication as the amount and probability of data loss is considerably lower, while performance remains the same. We present data from real production systems which shows that the dynamic system can maintain the required performance in the face of changes in the workload. We also show that in most cases the systems operate in non-synchronous mode for less than 10% of the time, thus lowering the exposure to a data loss event by a factor of at least 10, without sacrificing performance.

The paper is structured as follows. The next section presents some background on the Recover-Point system with its various synchronous and asynchronous replication modes. Section 3 discusses the method and algorithms of dynamic synchronous-asynchronous replication. In section 4 we consider some production case studies and analyze the behavior of the system based on real customer data. Section 5 describes related work, while section 6 summarizes our main conclusions and discusses future work.

2. THE RECOVER-POINT NETWORK BASED CONTINUOUS DATA REPLICATION

2.1. network based replication

In this section we describe Recover-Point, a network based continuous data protection and replication scheme. The Recover-Point replication system which features the dynamic synchronous-asynchronous replication is composed of two pieces, A *splitter* and a *data protection appliance*.

The splitter, intercepts the I/O arriving at the data path, and mirrors the I/O to both the network data protection appliance, and the original storage target. The splitter is installed either at the replicated host, or inside a fabric switch, or in a storage array. The splitter is a fairly simple and portable piece of code . The splitter can intercept data operations at several logical levels. The Recover-Point system implements a splitter for replication of block level devices which intercepts I/O at the `SCSI` layer.

The data protection appliance is a physical or virtual appliance, which exposes a SCSI target. It receives I/O coming from a splitter and replicates them over a WAN to a data protection appliance at the replica site. The replication appliance at the replica site manages the remote copy.

The replication appliance is a complex piece of code. The protocol between the replication appliances utilizes data compression and data de-duplication to optimize WAN traffic. For more on de-duplication methods, see [Zhu et al. 2008, Aronovich et al. 2009, Lillibridge et al. 2009]. The replica data protection appliance manages a *journal* of the data at the replica site as well as a full copy of the data.

The network based replication system, allows creation of consistency groups consisting of several volumes each. Since multiple splitters can send the I/O operations to the same data protection appliance, write order fidelity can be achieved between volumes from separate storage arrays at a single data protection appliance.

The journal is a log containing the sequence of changes that happened to each of the volumes in the consistency group. There are two streams of data in the journal. The first is a *redo log*, containing changes which occurred to the production volume and not yet applied to the replica volume. The second is an *undo log* containing a list of changes which undoes the latest changes in the replica volumes. Each entry in a journal contains, apart from the I/O data itself, the following I/O meta data:

1) A volume ID.
2) The I/O block offset within the volume.
3) The I/O length.

The meta data and the data of the journal may be stored in different streams. The meta data in the journal also contains information whether the point in time after the write described by the meta data is consistent. It may also contain some user created strings describing the point in time, e.g., a bookmark. The user may create a bookmark whenever desired. Each bookmark is a consistent point in time. If the user desires to create an application consistent point in time the user needs to place the application in a consistent mode such as Oracle hot backup mode or Microsoft VSS, and request the system to create a bookmark while the application is in the consistent mode. Creation of a bookmark is just a logical procedure which adds messages with the bookmark info into the I/O stream, and thus bookmark creation typically takes less than a millisecond. The full replica copy, together with the journal provides a Continuous Data Protection (CDP) system, allowing the user to access any point in time. Another example of a CDP system, which differs substantially from the one we implemented, is presented in [Shaull et al. 2008]. A system which offers multiple point in time views can also be found in [Strunk et al. 2000]. We now describe the operation of the system upon the arrival of a write I/O. The system is described in figure 1.

The I/O flow is as follows. A host application generates an I/O. The splitter, which is located somewhere in the data path, intercepts the I/O. The Splitter checks whether the I/O belongs to a volume which needs to be replicated. If the I/O should be replicated, the splitter mirrors the I/O to both the data protection appliance and the original I/O target, i.e., the production storage array. The splitter works at the SCSI protocol level,

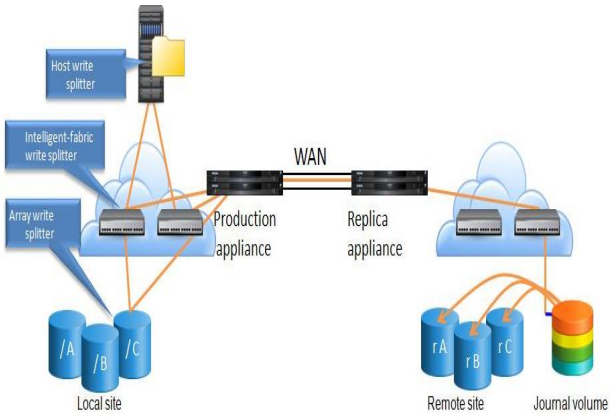

Fig. 1. Description of the system set-up.

and thus it intercepts SCSI write commands and sends the SCSI write command to a target exposed by the data protection appliance at the production site, and the original SCSI target of the command. The splitter returns the SCSI status of the write only after receiving status from both the original I/O target and the data protection appliance target. The protocol described above between the splitter and the data protection appliance basically states that the splitter mirrors the I/O synchronously between the production volume and the production site data protection appliance. The data protection appliance controls the I/O termination status. This allows the data protection appliance to implement any replication strategy desired. A synchronous replication strategy is implemented when the production appliance acknowledges the splitter I/O (by returning a SCSI status) after the I/O is sent to the replica site and an acknowledge is received from the replica site. We implemented two options for synchronous replication strategies. In one implementation the production appliance acknowledges the I/O to the splitter immediately after the I/O reaches the cache of the replica appliance. The I/O in this case is not yet safely written at the storage array at the replica site, but any disaster at the production site will induce no data loss as all the data is cached at the replication data protection appliance. The second implementation is synchronous replication to the replica site storage array. In this implementation the production appliance acknowledges the splitter only after the I/O is persistently stored in the journal at the replica site at the storage array.

An asynchronous replication strategy is implemented by decoupling the protocol between the local and remote data protection appliances from the acknowledgment to the splitter. The local data protection appliance immediately acknowledges the I/O to the splitter and only then the appliance asynchronously sends the I/O to the replica data protection appliance.

The asynchronous replication may be in one of two modes, a semi-synchronous mode, where all I/O are sent to the replica site, or a snapshot shipping mode, grouping large sets of changes together and removing *hot spots* from each changed set sent to the replica site. Hot spots are locations on a volume being replicated which are frequently accessed. When shipping a snapshot to a replica site we only need to transfer the latest data version which was written to a specific location on the storage device. If data was written to the same location on the storage device 10 times during the duration of the snapshot, the replication engine will only need to transfer one piece of data, saving

significant amounts of WAN traffic and also significant amounts of traffic to the replica storage array, [Patterson et al. 2002].

3. DYNAMIC SYNCHRONOUS/ASYNCHRONOUS REPLICATION

The system described above allows both synchronous and asynchronous replication. In this section we describe the methods and algorithms which dynamically decide when replication should be synchronous and when replication should be asynchronous.

The user configuration includes the following set of parameters:

1) Maximal allowed I/O write latency, which we denote by $MaxLat$.

2) Maximal RPO (recovery point objective) allowed, which we denote by $MaxRPO$. The maximal RPO is the maximal amount of data the user agrees to lose in order not to delay write I/O more than the maximum latency allowed. If the maximal RPO is set to infinity, the replication system will never delay writes by more than the maximal allowed I/O latency. If the maximal RPO is not set to infinity, delaying I/O for periods longer than the maximal allowed I/O latency will be allowed when the RPO becomes larger than the maximal RPO value. The system will try to remain synchronous as long as I/O latency can be maintained below the maximal allowed latency.

The system is built using a pair of algorithms. The first algorithm moves from synchronous replication to semi-synchronous and if needed to snapshot shipping replication as latency increases. The second algorithm moves from snapshot shipping replication to semi-synchronous replication and then to synchronous replication.

For each I/O arriving from the splitter to the data protection appliance the system creates the following data structure which describes the state of the I/O:

(1) I/O start time.
(2) transaction ID: Each I/O arriving at the data protection appliance gets a sequential transaction I/O number. Write order fidelity is achieved by handling the I/O in the order of their transaction ID.
(3) I/O transmission time: The time the I/O was sent from the production data protection appliance to the replica appliance.
(4) I/O acknowledge time: The time when it was acknowledged that the I/O reached the replica appliance.
(5) I/O journal flush time: The time an acknowledgment that the I/O was flushed to the journal arrived.

Some of the I/O statistics are used for modeling the IO average latency (e.g the start and acknowledge time), some are used for deciding the source of the latency (transition latency vs. latency from start time to acknowledge time). The transaction ID is used for write order fidelity, and the journal flush time is used to determine which data has been safely de-staged to the remote journal to allow freeing of resources.

3.1. Falling out of synchronous replication

When in synchronous replication, the system immediately sends I/O arriving from the splitter to the replica appliance, so the I/O transmission time is very close to the I/O start time. When the replica appliance acknowledges the I/O to the production data protection appliance, the production appliance acknowledges the I/O to the splitter and the I/O is completed by the host. The system continues to hold the I/O information even after the I/O was acknowledged to the host as the I/O may still be in the cache of the replica data protection appliance. Later when the I/O is flushed to the replica site journal, and released from the replica appliance cache, the journal flush time is

updated. When journal flush time arrives at the production site, the system state is evaluated and the I/O information may be removed from the I/O data structure. When in synchronous replication, the system monitors all the I/O for which acknowledge time is not updated. Since there is an ordering for the I/O imposed by the transaction ID counter, only the first non-acknowledged I/O needs to be evaluated. When the system observes that the average I/O latency for the last 2-5 seconds violates the latency policy, the system will change the state to asynchronous replication.

The asynchronous replication policy will be set to allow a lag of M megabytes. The value of M will be determined by the system in the following way:

The system considers all unacknowledged I/O. The system will count the amount of data in all the unacknowledged I/O which arrived over $MaxLat/2$ time units ago. The system immediately acknowledges all I/O which admit the newly set lag. The action of moving to asynchronous writing will immediately reduce the maximal latency of an I/O by a half, since all I/O which have been open for more than $MaxLat/2$ time will be immediately acknowledged. We note that in case of system failure these I/O will be lost. The system will continue evaluating the maximal allowed lag when in semi-synchronous replication. The maximal allowed lag value will be re-estimated depending on the maximal open I/O latency. If the latency reaches the $MaxLat$ value again, the value of M will be re-estimated and increased. If all unacknowledged I/O maintain a latency value smaller than $MaxLat/4$, the value M will be decremented. Using this simple feedback loop, the system tries to maintain a near optimal value of M which still satisfies the performance requirements. If the value of M grows beyond a certain amount, say $10MB$, the system will start merging multiple writes to I/O hot-spots in order to save I/O bandwidth.

It is important to note that when the system moves from synchronous replication to asynchronous replication, there is a de-coupling between the arrival of the I/O to the production appliance and the sending of the I/O to the replica appliance. As replication moves towards asynchronous replication, data optimization techniques, which save bandwidth but may add latency are added. These include data compression and de-duplication. If the maximal RPO configured by the user is limited, the value M will grow only until the limit that the user allowed. In this case the I/O latency may grow beyond the maximal value the user configured. If the user does not limit the maximal amount of data loss, the value of M may continue to grow to a point where the appliance cache will not be able to hold all the open I/O. In this case, the system will move to a complete snapshot shipping mode, reading the changes from the storage array.

There are two options for the snapshot shipping mode:

(1) *Track meta data mode*: The network based replication system is a heterogeneous replication system, as such, it cannot assume that production storage arrays implement snapshots. When the amount of data cached in the data protection appliance exceeds a certain amount the system cannot successfully send the data from the cache of the production appliance to the replica appliance. The system enters a high-load mode and discards any new incoming writes. Instead the system starts tracking the new writes in a production meta data journal. For each I/O arriving to the data protection appliance, the appliance will write the meta data of the write to a persistent store and discard the data of the write. We call this mode track meta data mode. Once the cache of the production data protection appliance is freed again, as data is sent to the replica site, the system will try to *synchronize* the volumes, i.e., send all the changed location tracks in the meta data stream to the replica site. To synchronize, the change list is read from the meta data stream, and data is read from the locations which are marked as changed and sent to the replica site. New I/O arriving from the splitter are also sent to the replica site and

the protocol ensures the write order fidelity between the splitter I/O and the I/O read from the production volumes.
The disadvantage of this method, is that the system will not be able to resynchronize the data, if the average amount of changes normalized by the compression ratio is larger than the available bandwidth over a substantial time period. The advantage is that it is completely asynchronous and storage agnostic.

(2) True snapshot shipping mode: Most storage arrays today support the creation of a snapshot. Integrating the networked based replication appliance with the array snapshot provides a snapshot shipping mechanism similar to that described in [Patterson et al. 2002]. The snapshot shipping mechanism consists of two components:
A snapshot, which creates a frozen copy of the storage volume at a point in time, and a *change tracker*. A change tracker is a mechanism which allows the system to get the list of changes between the last shipped snapshot and the next snapshot to be shipped to the replica site. The change tracker can be implemented within the storage array with different vendors providing different solutions to the tracking problem, [Hitz et al. 1994], [Patterson et al. 2002], [EMC Cellera], leveraging many of the same advantages as a Log Structured File System (LFS), [Rosenblum and Ousterhout 1992]. An API can provide the list of changes from the storage array to the network replication appliance. Another option is to implement the change tracking code within the data protection appliance. As described above, the appliance intercepts the I/O and can track the I/O meta data in a stream. When in snapshot shipping mode, the splitter does not actually need to send the data to the appliance and may send only meta data to the protection appliance.
After creating the snapshot at the production site, the system will read all the locations which are marked as changed, between the current snapshot and the last point in time shipped to the replica site as tracked in the change tracker, and transmit them to the replica site. Once the snapshot is fully shipped to the replica site, a new snapshot may be created, an so on.

3.2. Returning to synchronous replication

In a dynamic setting, the system will try to return to synchronous replication, if performance permits. Since synchronous replication increases the I/O latency, once the system falls out of synchronous replication it may wait a while before trying to return to synchronous replication to avoid I/O fluctuations. If the system reached snapshot shipping mode, it will first try to exit the snapshot shipping mode. The method of exiting snapshot shipping mode depends on the snapshot shipping mode we use.

(1) If the dynamic snapshot shipping mode is used, then, immediately when the system completes the synchronization of the volumes the system is no longer in snapshot shipping mode.
(2) In order to exit true snapshot shipping mode the change set between the current snapshot being shipped and the snapshot right after it must be smaller than the available cache in the production data protection appliance. When the system begins shipping a new snapshot it checks whether the size of the changes between the current snapshot and the previous snapshot is smaller than available resources. If so, the system will try to exit snapshot shipping mode. It will intercept all new I/O and hold them in the cache. Once the current snapshot is shipped, if the amount of changes did not exceed the size of the production appliance cache, the system will no longer be in snapshot shipping mode.

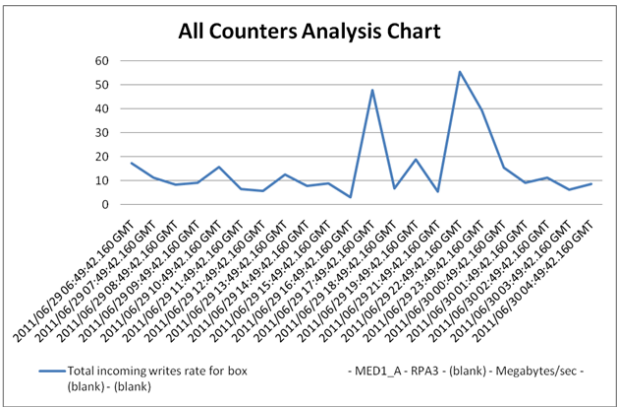

Fig. 2. Sample customer daily throughput (MB/sec) chart.

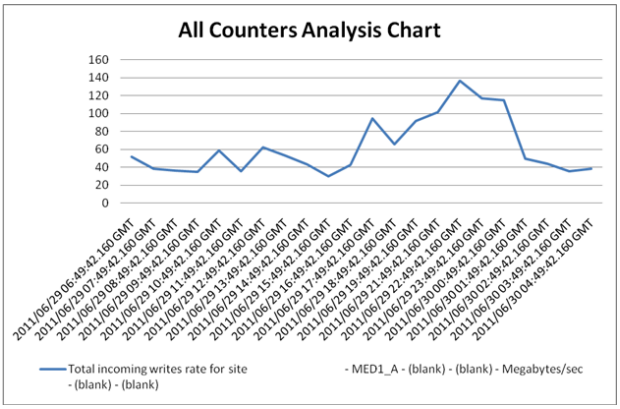

Fig. 3. Sample customer daily IOPS chart.

Once the system exits snapshot shipping mode, it is still not in synchronous mode. In fact, since the cache of the production appliance may be large it may still be far from being synchronous.

The system initiates a lag reduction procedure. The system will start to reduce the lag M, as described above, by decreasing the size of M whenever the I/O latency value is smaller than $MaxLat/4$. When the system is in a mode where is tries to return to synchronous replication it may also try a more aggressive strategy, decreasing the lag even when latency is $0.7 * MaxLat$. When the value of the lag M is small enough, say smaller than $0.5MB$, the system will move to synchronous replication, acknowledging new I/O only after they reached the replica site. The system will not allow frequent changes of state, so there will be a penalty for each time the system exits synchronous replication. Once the system exits synchronous replication, it will not try to return to synchronous replication for say, 5 minutes.

4. EXPERIMENTAL RESULTS

4.1. Typical work-flows

Figures 2 and 3 describe performance data gathered from a real customer production site. The graph in figure 2 describes the throughput at one data protection appliance. Figure 3 describes the number of I/O operations per second at the same appliance.

Between 21:00 PM and 1:00 AM there is a peak in the activity, reaching from 80MB/sec to 115MB/sec. The number of I/O operations per second is about 3800 I/O per second, the average I/O size is about 25KB. The data reflects data warehousing activity done at the customer site. The Customer data centers are 25 miles apart from each other.

We are going to use this production data in a controlled lab framework to show how the dynamic feature operates successfully in a real world situation, while tracking the main performance metrics. We are going to run a similar work flow, and test the system behavior when allowing a maximal latency of $4ms$.

4.2. Experimental setup

For hardware, we used two Recover-Point gateways with 2 GEN4 data protection appliances one appliance at each site. The interconnect between the two sites is a $4Gb$ fiber channel. The appliances each have $8192MB$ of RAM. Each appliance has a QLogic QLE2564 quad-port PCIe-to-8Gbps Fibre Channel Adapter. Each appliance in the pair is connected to a separate CLARiiON CX4-480 storage array with 30 FibreChannel-attached disks, configured as 6 separate RAID5, $4 + 1$ RAID groups. We configured a consistency group replicating 12 production volumes. The 12 production volumes were created on 4 separate RAID5 4+1 groups, 3 volumes were created on each RAID group. The journal was striped over two volumes from two separate RAID groups. We configured the replica site in an identical way. We use an ANUE fiber channel emulator to emulate a 25 mile distance WAN with limited 1Gbit bandwidth.

4.3. Performance metrics for synchronous and asynchronous replication

Figure 4 describes the I/O latency during asynchronous replication. The test does not limit the inter site bandwidth. We started by testing the latency of our asynchronous replication. The latency is measured from the host perspective and includes also the latency of the production storage array. The I/O were generated as small random I/O of size 8KB, but the I/O range was small enough, so that all the I/O would be write cache hits. This was done so that system performance will not be limited by the performance of the spindles. We configured the data protection appliance not to move to snapshot shipping mode and stay in semi-synchronous mode, so that hotspots will not reduce the I/O write traffic. The test used write I/Os only, to test the limits and effect of the replication system which only cares about write commands.

As can be observed, the per I/O latency is less than $1.5ms$ even at $10,000$ IOPS. At around $15,000$ IOPS, the latency starts to significantly increase as the load is about what one appliance can handle.

Figure 5 describes a similar test, this time using synchronous replication. Latency increases as the number of IOPS increases. At $6,000$ IOPS, the latency reaches $2ms$, at around $10,000$ IOPS, the latency reaches $3ms$ and at $15,000$ IOPS, the latency increases significantly.

As we can see synchronous replication has significant effect on performance even when there is no distance between the production and replica sites, compared to asynchronous replication.

Figure 6 describes a test where the WAN bandwidth was limited to 1Gb and a distance of 25 miles was emulated, as can be seen, the latency dramatically increases in such more realistic environments. There are several reasons for the latency increase. Synchronous replication over fiber channel uses SCSI write commands, each write performs 2 round trips and thus at a 25 mile distance the added latency is almost $1ms$. The limited bandwidth also increases the latency, since as the number of outstanding I/O grows, the amount of data needed to be sent before an I/O is acknowledged increases.

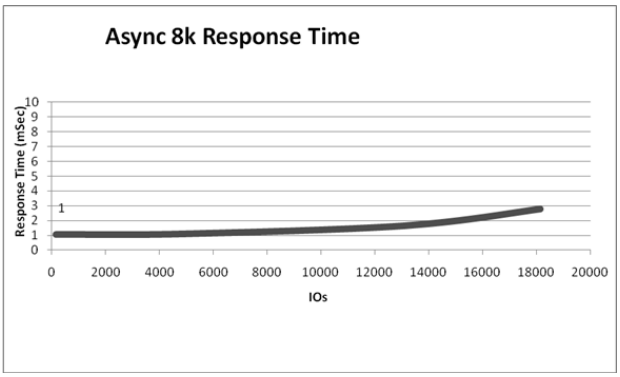

Fig. 4. Latency induced by asynchronous replication

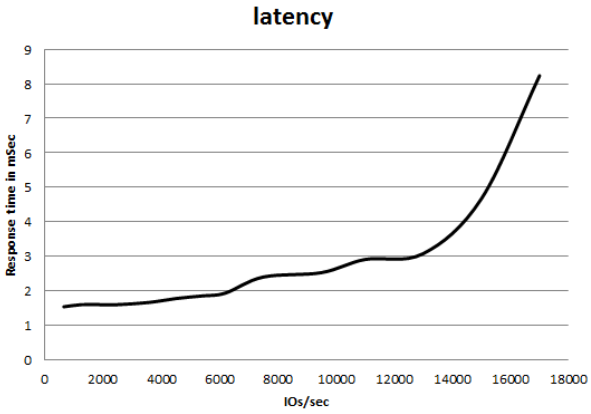

Fig. 5. Latency induced by synchronous replication, 8KB

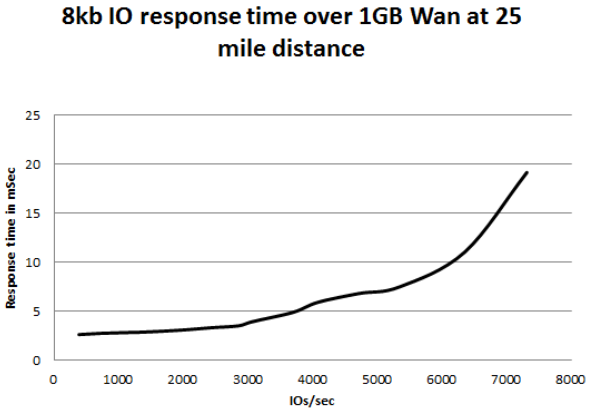

Fig. 6. Syncronous replication over 1GB WAN I/O and 25 mile distance

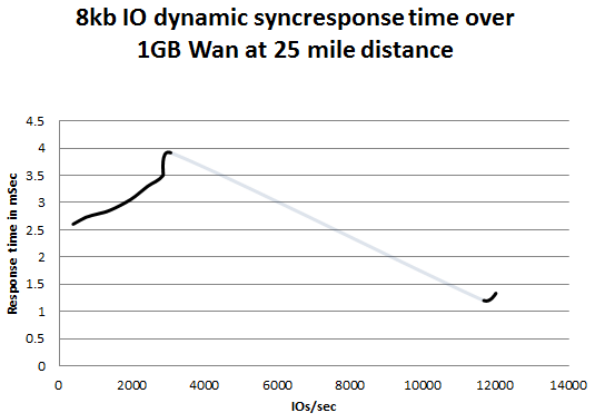

Fig. 7. Dynamic sync/async 8KB I/O, performance

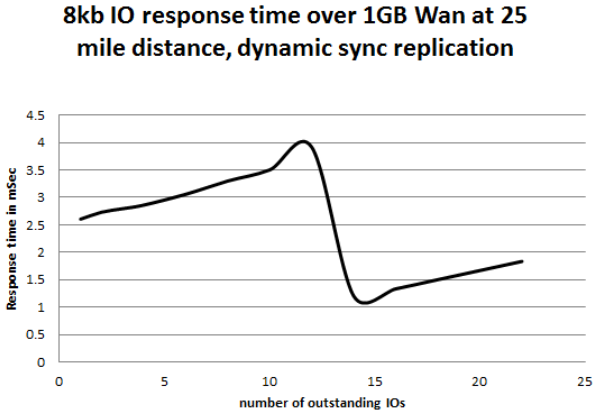

Fig. 8. Dynamic sync/async I/O latency/outstanding I/O

4.4. Dynamic synchronous asynchronous replication

We tested the system with the dynamic synchronous-asynchronous feature which is described in the paper. We configured the maximal latency to be $4ms$. We tested the system using IOMeter, an I/O subsystem measurement and characterization tool for single and clustered systems, [IOMeter]. We used an ANUE fiber channel emulator to emulate a fiber channle WAN with 1Gb of bandwidth and emulated distance of 25 miles. We tested the system, each time with a different number of outstanding I/O. When the number of outstanding I/O was around 12 the system was still in synchronous mode and the number of IOPS was $3,000$. The average I/O latency was almost 4 milliseconds. When the number of outstanding I/O reached 14, the system switched to asynchronous mode. The number of IOPS immediately jumped to almost 12,000 IOPS (which is very close to the available bandwidth), and the latency dropped dramatically to around $1.2ms$.

Figure 7 describes the I/O response time vs. the I/O rate, when using the dynamic synchronous-asynchronous feature.

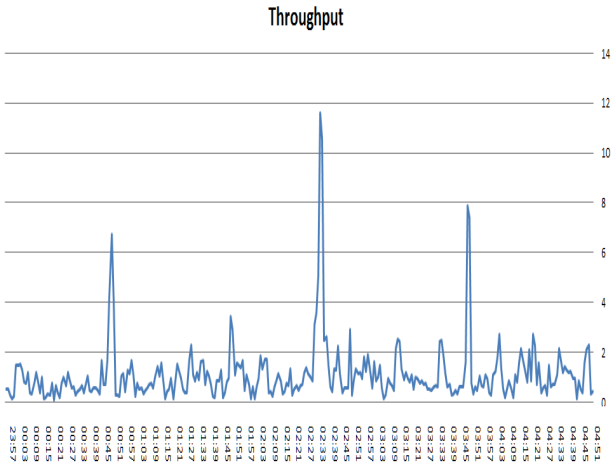

Fig. 9. Dynamic sync/async 8KB I/O, performance

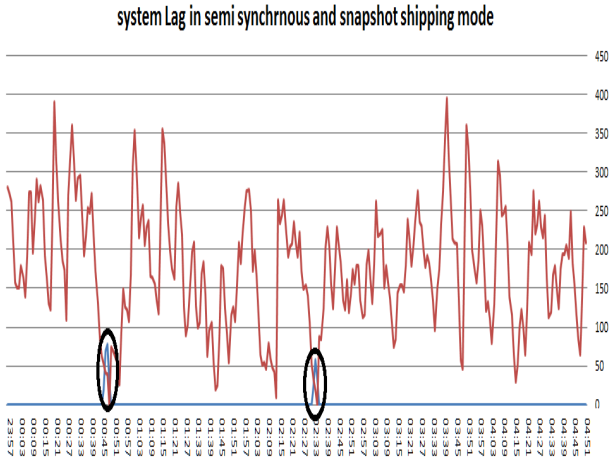

Fig. 10. lag between production and replica site (in MB)

Figure 8 describes the response time vs. the I/O latency when using the dynamic synchronous-asynchronous feature. As can be seen, the system moves to asynchronous replication when the average number of outstanding I/O was between 12 and 14.

4.5. Semi-synchronous to snapshot shipping replication

In the test performed above, the synchronous replication was implemented over a fiber channel connection. The latency limits were system limits and not channel throughput limits. In this section we show some customer production data, where semi-synchronous replication moves to the first mode of snapshot shipping described in section 3.1. Unlike the first example which was done in a lab, based on a customer profile, this example describes the performance of an actual customer production system.

The customer has a limited communication bandwidth of 2MB/sec. As can be seen, there are periods where load on the system is significantly higher than the channel bandwidth. For example, as described in figure 9, between 00:39 and 00:51 there is a peak of over 6 MB/sec. Between 02:27 and 02:39 there is a peak of over 11MB/sec and at 03:45 there is a peak of 8MB/sec. The customer above has a version of the data protection appliance with 1GB of cache available at the replication appliance. If the data lag is larger than 1GB, the cache buffer becomes full and the system will drop to snapshot shipping mode.

Figure 10 describes the lag (in MB) between the production site and the replica site (the red line). The lag was only measured when the system did not move to type 1 snapshot shipping mode. The blue lines (surrounded by circles) describe the times when the system is in snapshot shipping mode. As we can see this event occurs at time intervals 00:39-00:45 and 02:29-02:37. During those time intervals, the system could not handle the incoming load due to the low WAN bandwidth available and went into snapshot shipping mode. At 03:45 the system was able to handle the higher load but the lag grew to 400 seconds.

The results for full snapshot shipping mode are very similar to the results in [Patterson et al. 2002] Table 1, in terms of the amount of bandwidth required for the differentials.

4.6. expected behavior of the system

We analyzed real production data which was gathered from 20 customers, running over 50 replication appliances, replicating hundreds of consistency groups. Each replication appliance gathered detailed I/O traces for a period of between one and two weeks. We than ran a simulation that computed the percentage of time that the appliance will be in synchronous replication mode as compared to asynchronous replication assuming average requested response times of 4,5 and 7 ms respectively. Figure 11 describes the system behavior when running dynamic synchronous replication over a distance of 25 miles using a 1gb WAN per replication appliance. As we can seen, even at a 4ms response time, 15 of the 20 customers remain in synchronous replication mode over 90% of the time. This produces a 10 fold reduction in exposure to data loss in comparison with asynchronous mode without sacrificing any performance. At 7 ms response time about half of the customer remain in synchronous mode at all times, except very rare cases which occur less than 0.1% of the time. Without the dynamic feature, these systems would be exposed to data loss at all times, just to protect performance on these very rare occasions. We further note that manual adjustment between the two modes is very difficult to perform and is inefficient since it is difficult in many cases to predict when performance issues will occur. Thus the solution must be automated as presented in this paper.

Figure 12 describes the expected system behavior when running dynamic synchronous replication over a distance of 25 miles using a 2gb WAN per replication appliance. In this case, a vast majority of the customers remains in sync almost all the time. There are customers though which will still remain in asynchronous replication for over 10% of the time.

Customers typically balance their activity over several replication appliance since each replication appliance has bounded performance. As the bandwidth grows, more customers will remain in synchronous replication for longer periods of time.

5. RELATED WORK

There are several types of on-line data replication techniques. Data replication techniques typically come in one of two flavors, either host based or array based. The `Coda` system is one early example of a replicated file system, [Kistler 1993]. In Coda, the

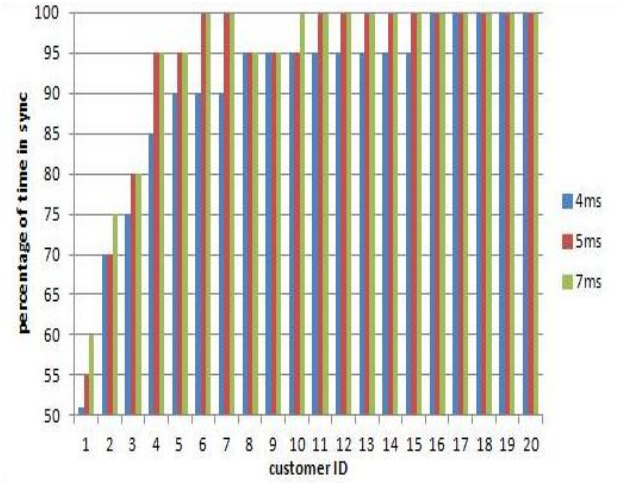

Fig. 11. expected time in synchronous replication over 1gb link

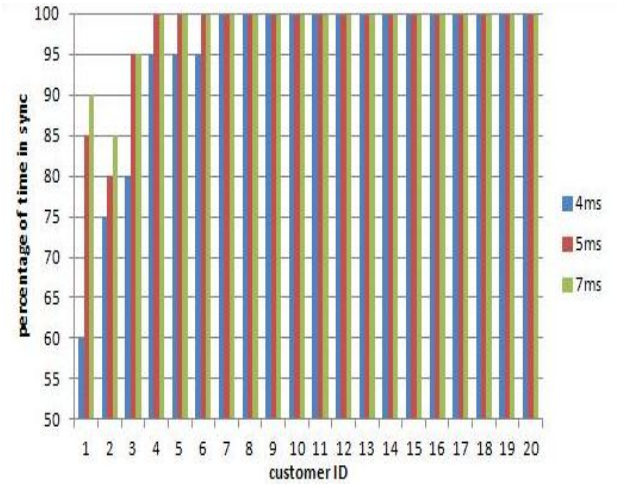

Fig. 12. expected time in synchronous replication over 2gb link

clients of a file server are responsible for writing to multiple servers. This approach is essentially synchronous logical-level mirroring. By putting the responsibility for replication on the clients, Coda effectively off-loads the servers, and because clients are aware of the multiple servers, recovery from the loss of a server is essentially instantaneous. However, Coda is not designed for replication over a WAN. If the WAN connecting a client to a remote server is slow or congested, the client will feel a significant performance impact.

Array based replication includes both synchronous replication and asynchronous replication technologies. For a nice introduction to remote mirroring techniques see, [Ji et al. 2003]. Efficient asynchronous replication schemes are described in [Weatherspoon et al. 2009] and [Yan et al. 2004]. For descriptions of commonly available commercial implementations, which have both synchronous and asynchronous modes, see, [Patterson et al. 2002] and [Azagury et al. 2003]. Other methods for optimizing replica-

tion can be found in [Weatherspoon et al. 2009]. However, none of these systems moves dynamically between the two basic modes.

At the logical file system level, there have been several file systems which replicated data to improve reliability, [Liskov 1991], [Rosenblum and Ousterhout 1992], [Matthews 1997], [Ghemawat et al. 2003]. The most common approach is to scan the directory structure checking the time that files were last updated. For example, the UNIX dump utility compares the file modify times to the time of the last dump to determine which files it should write to an incremental dump tape. An example of a method of detecting new data at the logical level includes programs like rdist and rsync, [Tridgell and Mackerras 1996]. These programs traverse both the source and destination file systems, looking for files that have been more recently modified on the source than the destination. The rdist program will only transfer whole files. If one byte has changed in a large database file, the entire file will be transferred. The rsync program works to compute a minimal range of bytes that need to be transferred by comparing checksums of byte ranges. Another example of an asynchronous replication scheme is presented in [Leung et al. 2002].

For a general discussion of disaster recovery in storage systems, see [Keeton et al. 2004].

6. CONCLUSION AND FUTURE WORK

Current techniques for disaster recovery offer the user a choice between synchronous and asynchronous replication. This requires customers to make a hard choice between reasonable system performance and significant risk of data loss. In the paper we presented the first replication system that allows customers to enjoy both good performance and to minimize the risk of data loss. This is done by dynamically moving between the modes of synchronous replication and asynchronous replication. Users may prefer synchronous replication but may have a higher priority for not delaying I/O operation more than a specific amount of time. Synchronous replication may lower the maximum throughput and number of IOPS of the user and may cause batch operations and data warehousing operations to take significantly longer. Allowing the system to temporarily move out of synchronous replication improves the performance of batch operations significantly.

In the future we may add a scheduler which will prevent the system from exiting synchronous replication, for instance, during working hours. This feature will allow the system to dynamically move to asynchronous replication only during non-working hours where information arriving to the system may be less significant. Another area for future research is on how to reduce the lag, when we desire to move from asynchronous replication to synchronous replication effectively, with minimal delay for I/O. The current system implementation measures the latency from the time an I/O reaches the production data protection appliance. In a future implementation the I/O latency may be measured at the production splitter and may also include, depending on the configuration, the latency between the splitter and the production data protection appliance and then also the production storage latency. This will provide a more accurate measure of I/O latency as seen from the point of view of the user. We also plan to add a simple economic model for the penalty involved in data loss that will help customers balance between performance and risk of data loss.

7. ACKNOWLEDGMENTS

We thank Ariel Kulik, Lev Ayzenberg and Ido Singer for their help in implementing the Synchronous/Asynchrnous replication mechanism, and Moshe Hermos, for help with the performance testing.

REFERENCES

L. Aronovich, R. Asher, E. Bachmat, H. Bitner, M. Hirsch, S.T. Klein: The design of a similarity based deduplication system. SYSTOR 2009: 6.

Azagury A., Factor M. and Micka W. 2003. Advanced functions for storage subsystems: Supporting continuous availability, *IBM SYSTEM Journal*.

EMC Celerra Replicator http://www.emc.com/.

EMC Symmetrix Remote Data Facility. http://www.emc.com/.

Ghemawat S., Gobioff H. and Leung S. 2003. The Google file system, *Proceedings of ACM SOSP*, 29-43.

Hitz D., Lau J., Malcolm M.A. 1994. File System Design for an NFS File Server Appliance, *Proceedings USENIX Winter Conference*, 235-246.

IOMeter http://www.iometer.org/

Ji M., Veitch A. and Wilkes J. 2003. Seneca: remote mirroring done write, *Proceedings of USENIX Technical Conference (San Antonio, TX)*, 253-268, June 2003.

Keeton K., Santos C., Beyer D., Chase J. and Wilkes J. 2004. Designing for disasters. *Proceedings of USENIX FAST, USENIX Association*, pp. 59-62.

Kistler J.J. 1993. Disconnected Operation in a Distributed File System, *Technical Report CMU-CS- 93-156. School of Computer Science, Carnegie Mellon University*, 1993.

Leung S.A., Maccormick J., Perl S.E. and Zhang L. 2002, Myriad: Cost-effective disaster tolerance, *Proceedings of USENIX FAST*.

Lillibridge M., Eshghi K., Bhagwat D., Deolalikar V., Trezise G., and Camble P. 2009. Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality, *Proceedings of USENIX File And Storage Systems conference (FAST)*.

Liskov B., Ghemawat S., Gruber R., Johnson P., Shrira L., and Williams M. 1991. Replication in the Harp file system, *Proceedings of 13th SOSP, published as Operating Systems Review*, 25(5), 226-238.

Matthews J., Roselli D., Costell A., Wang, R., and Anderson, T. 1997. Improving the performance of log-structured file systems with adaptive methods, *Proceedings of ACM SOSP*.

Patterson R.H., Manley S., Federwisch M., Hitz D., Kleiman S. and Owara S. 2002. SnapMirror: File-System-Based Asynchronous Mirroring for Disaster Recovery, *Proceedings of FAST 02, the 1th USENIX Conf. on File and Storage Technologies*, 117-129.

Rosenblum M., Osterhout J.K. 1992. The Design and Implementation of a Log-structured File System, *ACM Transactions on Computer Systems*, Vol.10, No.1, 26-52.

S. Savage and J. Wilkes, AFRAIDA Frequently Redundant Array of Independent Disks. In Proc. Winter USENIX, (San Diego, CA) pages 2739, Jan. 1996. USENIX.

Shaull R., Shrira L. and Hao X. 2008. Skippy: a New Indexing Method for Long-Lived Snapshots in the Storage Manager, *ACM SIGMOD Conference 2008*.

Strunk J.D., Goodson G.R., Scheinholtz M.L., Soules C. and Ganger G.R. 2000. Self-securing storage: protecting data in compromised systems, *Proceedings of the 4th OSDI*, 165-180.

Tridgell A. and Mackerras P. 1996. The rsync algorithm, *Department of Computer Science Australian National University, TR-CS-96-05*.

Weatherspoon H., Ganesh L., Marian T., Balakrishnan M., and Birman K. 2009. Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance, *Proceedings of FAST 09, the 7th USENIX Conf. on File and Storage Technologies*, 211-224.

Yan R., Shu J. and Chan Wen D. 2004. An implementation of semi-synchronous remote mirroring system for SANs, *In ACM Workshop of Grid and Cooperative Computing (GCC)*.

Zhu B., Li K. and Patterson H. 2008. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System, *Proceedings of FAST 08, the 6th USENIX Conf. on File and Storage Technologies*, 279-292.