Sun StorEdge™ Availability Suite 3.0 and 3.1 Software — Using The Remote Mirror Software In Asynchronous Replication Mode

A Best Practice

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 817-0055-10
January 2003, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

Remote Mirror Software Asynchronous Mode

This best practice document describes the advantages of using the Sun StorEdge™ Availability Suite remote mirror software in asynchronous mode to replicate data over a highly-latent network link. (This software also operates in synchronous mode.) This document describes the following:

- Factors that influence the overall application performance
- How to configure the software to help achieve the best results when using asynchronous mode replication

For any enterprise that values information as a critical business asset, business continuity is a vital issue. Any business continuity planning must consider the impact of down time and how to implement solutions required to minimize the risk of data loss. The Sun StorEdge Availability Suite software helps provide a foundation for building sound business continuity plans that offer continuous access to applications and data in the event of unplanned downtime or disasters.

Remote Mirror Software Replication Modes

The remote mirror software helps protect customer information assets from unplanned down time and disasters by replicating data between physically-separate hosts – in real time and from any distance. This software helps reduce disruptions to critical business data, helps support business continuity planning, and helps to establish rapid disaster recovery. The remote mirror software can use any type of IP-based network interface to replicate data, including ATM, ISDN, Ethernet, Gigabit Ethernet, T1, and T3.

It can operate in two replication modes: synchronous and asynchronous.

Note – As no replication occurs during application read operations, this document describes application write operations only. The *Sun StorEdge Availability Suite 3.1 Remote Mirror Software Administration and Operations Guide* describes the software and its features.

Synchronous Mode

In synchronous mode, the write data is committed to storage on the primary and secondary hosts before operation completion is acknowledged to the application. That is, the write operation is not deemed complete until the acknowledgement is received from both the primary and the secondary hosts, which helps to ensure that no data is lost in case of failure during the write.

As long as the link remains active and open, the secondary host has the same information as the primary host at any given time to help ensure faster fail over or recovery scenarios.

Because this replication type is affected by network latency and bandwidth, high latency or narrow bandwidth might cause the application to experience performance degradation. You can implement this replication mode on a low latency, high bandwidth environment as it can provide full data currency across multiple hosts.

Asynchronous Mode

In asynchronous mode, the application write operation is written to the primary volume and also to the local asynchronous queue. One queue is maintained per remote mirror volume set (if the volumes are not grouped) or per group (if the volumes are grouped). The application write operation is acknowledged as completed as soon as:

1. The data is placed in the queue
2. The bitmap for the data block is set

At a later time, the data from the queue is replicated to the secondary host when network bandwidth allows. Once the write operation acknowledgement is received from the secondary host, the data is removed from the queue.

This mode can potentially remove the network latency overhead from the application while providing real-time response times and updates.

Note – When replicating in asynchronous mode, to help preserve write ordering across the secondary host volumes, group the volumes and replicate them in this mode. When the volumes are grouped, the remote mirror software uses one queue per group instead of maintaining one queue per volume.

Comparing Asynchronous and Synchronous Modes

TABLE 1 compares the two replication modes.

TABLE 1 Replication Modes Compared

Synchronous Replication	Asynchronous Replication
A write operation is not complete until that write is posted to the secondary host.	The write operation is complete as soon as the data is placed in the asynchronous queue.
The response time depends on the network.	The response time does not depend on the network, as long as the queue is not filled. Can be a faster response time than synchronous replication.
Because every data change is replicated to the secondary host in real-time, the secondary host is in sync with the primary host.	The secondary host lags behind the primary host, depending on the latency, queue size, and the volume of writes.
Write ordering across volumes is preserved at the secondary host.	If write ordering is required across volumes at the secondary host, then group the volumes at the primary host.
This mode is meant for the data volumes that have zero tolerance of data loss at the secondary host.	This mode is meant for data volumes that have some level of tolerance for data loss at the secondary host.
When multiple threads of writes occur, all threads are replicated to the secondary host in parallel.	Multiple thread writes are placed in the queue and one flusher thread reads from the queue and replicates to the secondary host.

Factors That Might Affect Application Performance

The following factors might affect or otherwise impact your application's performance when using the remote mirror software in asynchronous mode.

Response Time

Response time is the amount of time required for an application to complete a particular I/O operation. With a longer response time, an application can perform fewer I/O operations per second than with a shorter response time. Among the factors influencing response time are:

- Network latency
- CPU type and speed
- Memory size and usage
- Storage type
- Data layout
- Read and write latency

Each factor might become a bottleneck, affecting response time and degrading application performance.

Data Type (Raw Volume or File System)

Because the remote mirror software is a volume replicator, and not a file system replicator, it replicates volumes or devices to the secondary host. The software does not have knowledge of the data type on storage devices. The device can be a raw device or it can contain a mounted file system.

Raw Volumes	These volume types use character-mode nonbuffered interfaces and perform I/O operations between the application buffer and the physical device. The write operations are characteristically synchronous.
File System Volumes	File systems such as UFS, VERITAS (VxFS), Sun™ QFS, and Sun SAM-FS use their own buffer to capture the write operations from the application. The writes are flushed to disk later. The writes are already asynchronous in nature. However, these file systems have the overhead of maintaining metadata. All file systems have performance advantages and disadvantages.

Network Interface and Bandwidth

The remote mirror software replicates over any IP-based network interface link. Each interface has its own bandwidth limitation. The bandwidth available for a data transfer depends on network utilization. If multiple applications share the same network, the bandwidth available to the remote mirror software might be reduced.

The table shows some of the network interface types and the amount of bandwidth each can theoretically handle.

Network Link	Theoretical Bandwidth
Gigabit Ethernet	125 Mbytes per second
ATM622	75 MBytes per second
100BaseT	12.5 MBytes per second (100BaseT is the most common network interface type)

TCP Buffer Size

The TCP buffer size is the number of bytes that the transfer control protocol allows to be transferred before it waits for an acknowledgement. See Appendix B for details.

Network Latency

Network latency is the amount of time it takes for a data packet to perform a round trip from one host to another. Higher latency means that the round trip is usually longer. The latency affects the time required to perform a network data transfer.

Network Throughput

Network throughput is the amount of data that can be transferred across the network per second. I/O operations (IOPs) occurring to the volume are computed as `1/(latency in seconds)`. Typically, 20 IOPs per second can be performed on a volume with network latency of 50 milliseconds.

For example, with a write size of 8 Kbytes, the network throughput is 160 Kbytes per second, or `8 Kbytes*20`.

Number of Volumes or Threads

Parallelism can increase the network utilization up to the point where other factors limiting scalability prevail. To achieve this parallelism, using multiple volumes, instead of perhaps one volume, can help increase remote mirror throughput.

Other Factors Affecting Performance

Queue Size	Choosing the proper queue size is critical for the best performance of asynchronous replication. In asynchronous mode, the application receives acknowledgement of a write operation when the data is placed in the asynchronous queue. If the asynchronous queue is full, subsequent write operations have to wait to be placed in the queue. As a result, the application response time increases, which is not desirable. To help improve the response time for the application, increase the asynchronous queue size based on its usage. See Appendix A for information about tuning the queue size.
Fill Rate	Fill rate is the rate at which the application writes are being filled in the asynchronous queue. It is directly proportional to the application throughput, which is IOPs times the transfer size. See Appendix A.
Drain Rate	The drain rate is the rate at which the data is replicated and removed from the asynchronous queue. The remote mirror software version 3.1 as tested can perform a maximum drain of 32 Kbytes per kernel remote procedure call (kRPC). If the data queue has one data packet up to 32 Kbytes in size, it is replicated with one kRPC. If the packet size is more than 32 Kbytes, the software performs multiple kRPCs. After the write is completed at the secondary host, the data is removed from the queue. With higher latency, the drain rate would be slower. See Appendix A.
Async Queue Memory Usage	The asynchronous queues are allocated in kernel memory. The memory consumption is calculated by adding up all the queue sizes. If more memory is consumed just by the queues, it adversely affects the overall performance of the system. Using too much kernel memory for asynchronous queues might adversely affect overall system performance.

Replicating Over A High-Latency Link Using the Point-in-time Copy Software

On a highly-latent network, synchronous mode replication might adversely affect the application performance. Asynchronous mode decouples application writes from network write latency as long as the network queues do not reach their full point. The queues exist to absorb bursts of application writes.

When the asynchronous queue for a volume is full, the application writes must wait for room in the queue, which increases the response time to that of synchronous replication. If the application often has a write rate that exceeds the drain rate for periods of time long enough to fill the queues, consider using the Sun StorEdge Availability Suite point-in-time copy software in conjunction with the remote mirror software.

To use the point-in-time software in this case, periodically make a shadow volume copy of the master primary volumes and then replicate the shadow volume to the secondary host.

In this case, one advantage is that you are removing any network latency and queue bottlenecks from the application. Another advantage is that you maintain a second copy of your data at the primary host, which can be used in case of some primary volume or host failure. However, the data at the secondary host lags behind the primary host data depending on the frequency of the copy and replication operations.

For more information, see the *Sun StorEdge Availability Suite 3.1 Point-in-time Copy Software Administration and Operations Guide* and the best practice document *Sun StorEdge Availability Suite Software — Improving Data Replication Over a Highly Latent Network*, part number 816-7180.

Suggested Replication Modes for Use with Databases

The database environment is broadly classified into the following categories:

- OLTP (Online transaction processing) – typically transactions have an 8 Kbyte block size and are random in nature
- DSS/DWH (Decision Support System/Data Warehouse) – typically transactions have a greater than 16 Kbyte block size and are mostly sequential in nature

TABLE 2 Suggested Replication Modes Depending on Database Use or Workload

Database Application	Workload Situation	Latency between Hosts	Selected Data Files	Replication Mode
OLTP	Light	High	All	Asynchronous or see "Replicating Over A High-Latency Link Using the Point-in-time Copy Software" on page 9
OLTP	Light or heavy	Low	All	Synchronous
OLTP	Heavy	High	All	See "Replicating Over A High-Latency Link Using the Point-in-time Copy Software" on page 9
			Redo Logs (have a standby database at the secondary host)	Synchronous or asynchronous
DSS/DWH	Data is being loaded	Any	All	Logging
DSS/DWH	Completed and read only	High	All	Asynchronous
DSS/DWH	Completed and fewer writes	Low	All	Synchronous
Import or tape restore	Any load	Any	Selected volumes	Logging

Note – Whether a particular network is experiencing high or low latency must be decided by the user. For instance, a user might consider low latency as less than 10 milliseconds and anything higher as a highly latent link.

Performance Observed

Performance tests using the remote mirror version 3.1 software were executed between primary and secondary hosts having a network latency of 50 milliseconds. Identical tests were performed in both synchronous and asynchronous modes. Both the primary and secondary hosts included the following specifications:

TABLE 3 Systems Used

Machine Type	Sun Enterprise™ 4500 server with 4 (four) CPUs and 4 (four) Gbytes of memory
Latency	50 milliseconds
Link Type	100 Mbit link
Remote Mirror Software Version	3.1
TCP Buffer Size	356 Kbytes
Number of Raw Volumes	5 (five)
Queue Size	8 Mbytes

TABLE 4 and FIGURE 1 show the advantage of using the asynchronous replication mode over a high latency link. For different types of I/O operation, the response time is faster in asynchronous mode as compared to the synchronous replication mode.

Note – In TABLE 4, the asynchronous mode response time is the maximum time of a write to queue or a write to local disk.

TABLE 4 Asynchronous Mode Performance Results

Block Sizes	Write Type	Version 3.1 Synchronous Response Time	Version 3.1 Asynchronous Response Time	Improvement Percent
8 Kbytes	Sequential	60 milliseconds	10 milliseconds	500
32 Kbytes	Sequential	66 milliseconds	18 milliseconds	270
64 Kbytes	Sequential	124 milliseconds	25 milliseconds	395

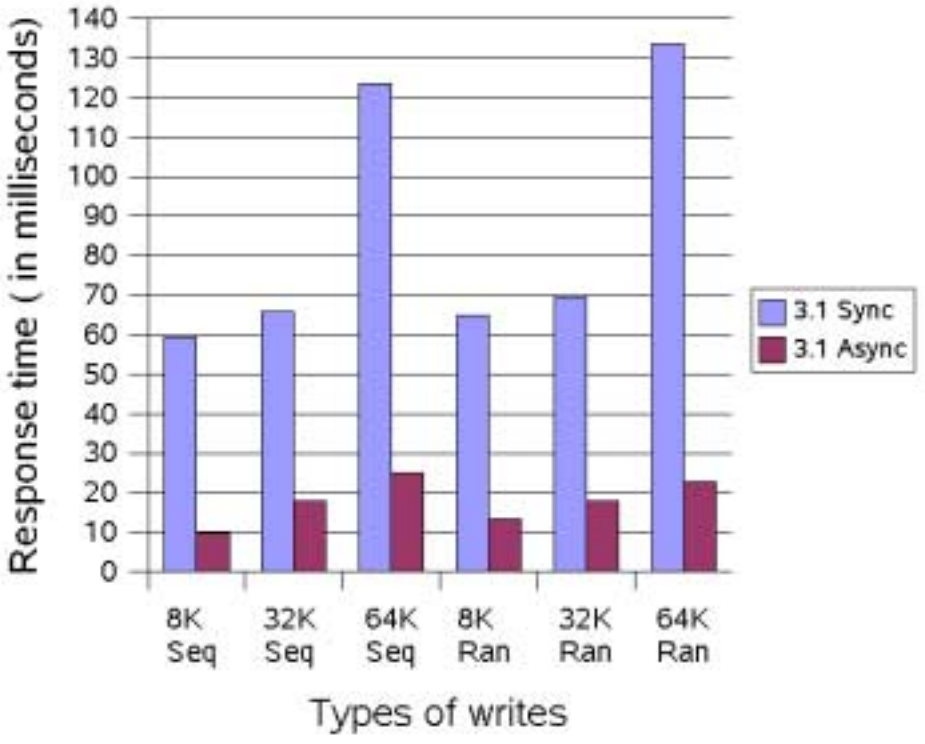

FIGURE 1 Sequential and Random Write Response

- As far as the file system is concerned, since the data is buffered before writing to the disk (if direct I/O is not forced), the writes are already in asynchronous mode. Note that when the data is flushed by the file system, the data enters the I/O stack and through to the remote mirror software to the underlying disk or volume manager.
- The response time in asynchronous mode is the maximum of write to queue or write to local disk.
- Using cached arrays (such as the Sun StorEdge T3 array) helps improve response time when compared to using JBODs (“just a box of disks”). The improvement occurs because the write operation is acknowledged as soon as the data reaches the array cache. Also disk writing latency is not involved.
- For a 0 millisecond latency link, using asynchronous mode does not yield additional performance improvements. Rather, it imposes the queue size and single thread replicating issues.

Summary

- On a high latency link, asynchronous replication can improve service times when compared to synchronous replication, as long as the asynchronous queue is not filled.
- On a low latency link, synchronous mode of replication might be the best mode to choose.
- For write-intensive workloads on a high latency link, if both synchronous and asynchronous replication modes affect application performance, consider using the point-in-time copy software to copy the production master volume and replicate the shadow copy. The frequency of copy and replication is based on the tolerance level of data loss at the secondary host.
- If a burst of activity is observed during a certain period of time (such as batch updates and heavy transactions for short periods of time), then it is recommended that you place the volumes into logging mode. The data can be fast-synchronized with the remote host later. However, note the implications of losing data in case of a disaster.
- In asynchronous mode, the bigger the queue, the more transactions the application can perform without filling up the queue. However, as a boundary exists, the queue might eventually fill if the rate of writes continues. This queue fill could eventually cause the secondary host to become further out of sync.

Conclusion

Use the asynchronous replication mode if the application or user requires a faster response time over a highly-latent network link.

However, you must consider the following:

- Fill rate - Too much updated data combined with a lower drain rate might fill the queue and slow down the application.
- Sizing of the asynchronous queue - The size would be the maximum amount of data which the secondary host volume lags behind the primary host. (That is, the queue size would equal the amount of primary host data yet to be replicated to the secondary host.)

APPENDIX **A**

Asynchronous Queue

In asynchronous mode, the data to be replicated to the secondary host is first placed in the asynchronous queue. This appendix describes how to discover the queue size and tune it. The following topics are described:

The *Sun StorEdge Availability Suite 3.1 Remote Mirror Software Administration and Operations Guide* also describes how to tune the queue size.

Queue Size

In the Sun StorEdge Availability Suite version 3.0 (Sun StorEdge Network Data Replicator 3.0 or 3.0.1), the default size of the asynchronous queue is 60 write operations and 800 512-byte blocks (or FBAs). With these sizes, it is easy to hit the upper limits of the queue (roughly 400 Kbytes of data) before the queue would start to affect application performance. The queue is allocated within the storage device block cache (SDBC) layer and can become a bottleneck for normal read and write operations.

In the Sun StorEdge Availability Suite version 3.1 remote mirror software, the queue characteristics are as follows. Depending on the application write operations, you can tune the queue size.

Default maximum number of write operations in the queue (tunable)	4194304
Default maximum number of 512-byte data blocks (default queue size) (tunable)	16384 (about 8 Mbytes of data)
Maximum queue drain rate (not tunable)	32 Kbytes per kRPC

For version 3.1, the queue is allocated in kernel anonymous buffers (KMEM allocation) and reduces the load on the SDBC. Also, the queue size can be increased depending on the availability and usage of kernel memory.

The asynchronous queue should be monitored using the `kstat`(1M) command. Check for the high water mark (HWM) for queue utilization on a regular basis. If the HWM (where the application writes more data than the queue can handle) frequently reaches 80 to 85 percent, then increase the queue size.

For example, for the default queue size of 16384 blocks, check that the high-water mark does not exceed 13000 to 14000 blocks. If it exceeds this amount:

1. Place the volume into logging mode (using the `sndradm -l` command).
2. Resize the queue size using the `sndradm -F` command.
3. Perform an update synchronization by using the `sndradm -u` command.

▼ To Display the Current Queue Size

1. **Type the following to display the current queue size:**

```
# sndradm -P

/dev/vx/rdsk/rootdg/ds4-clone   ->      nws:/dev/vx/rdsk/rootdg/ds4-sndr-s
autosync: on, max q writes: 4194304, max q fbas: 16384, mode: async
state: replicating
```

The `kstat` command also displays queue information:

2. **Perform any of the following `kstat`(1M) commands to show more information.**
   - To show all sets, type:

```
# kstat sndr:::maxqfbas
```

   - To show the first instance (0) in the queue, type:

```
# kstat sndr:0::maxqfbas
```

   - To show more information:

```
# kstat sndr:::maxqitems
# kstat sndr:::async_throttle_delay
```

Sample kstat Output For a Correctly-Sized Queue

Note – This example shows only a portion of the command output required for this section; the kstat command actually displays more information.

The following kstat(1M) kernel statistics output shows information about the asynchronous queue. In this example, the queue is sized correctly and not currently filled.

```
# kstat sndr:0:setinfo
module: sndr                            instance: 0
name:   setinfo                         class:    storedge
        async_block_hwm                 878
        async_item_hwm                  483
        async_throttle_delay            0
        maxqfbas                        16384
        maxqitems                       4194304
        primary_host                    regina
        primary_vol                     rootdg/ds4-clone
```

This example shows the following settings and statistics:

maxqfbas 16384	Default maximum number of 512-byte blocks in the asynchronous queue. Set this using by sndradm -F The default setting allows about 8 Mbytes of data per queue.
maxqitems 4194304	Default maximum number of write operations to queue for asynchronous mode volume sets. Set this by using sndradm -W. According to this setting, each write consumes 2 Kbytes for the 8 Mbytes of data.
async_block_hwm 878	[1] A total of 878 512-bytes blocks (approximately 439 Kbytes) have been put into the queue.
async_item_hwm 483	[1] A total of 483 write transactions have been put into the queue.
async_throttle_delay 0	No delay in the queue; that is, the queue is not yet full.

1 async_block_hwm and async_item_hwm show the maximum number that have been put into the queue since replication started. They do not show the current number that are in the queue.

Sample `kstat` Output for an Incorrectly-Sized Queue

Note – This example shows only a portion of the command output needed for this section; the `kstat` command actually displays more information.

The following `kstat`(1M) kernel statistics output shows information about the asynchronous queue, which is incorrectly sized.

```
# kstat sndr:4:setinfo
module: sndr                            instance: 4
name:   setinfo                          class:    storedge
        async_block_hwm                 16380
        async_item_hwm                  2045
        async_throttle_delay            16497
        maxqfbas                        16384
        maxqitems                       4194304
        primary_host                    andrea
        primary_vol                     rootdg/ds-forall
```

This example shows the default queue settings but the application is writing more data than the queue can handle. The `async_block_hwm` value of 16380 indicates that the application is approaching the maximum allowed limit of 512-bytes blocks. It is possible that the next few I/O operations are not going to be placed into the queue.

The value of `async_throttle_delay` indicates that the application writes have waited 16497 x 2 milliseconds to enter the queue. Increasing the queue size using the `maxqfbas` parameter might be a solution in this case. However, consider improving the network link (such as using larger bandwidth interfaces) to achieve long-term benefits. Alternatively, consider taking point-in-time volume copies and replicating the shadow volumes. See the *Sun StorEdge Availability Suite 3.1 Point-in-time Copy Software Administration and Operations Guide.*

Tuning the Asynchronous Queue

For asynchronous queue tuning, identify the volumes that experience more write operations. Since the asynchronous queues are maintained in kernel memory, increasing the queue sizes of all the volumes will cause overall system performance degradation.

The following information is needed on a per volume basis to tune the asynchronous queue size.

- Average writes per second (obtained by using the `iostat`(1M) command)
- Block size of writes (8 Kbytes, 16 Kbytes, and so on)
- Network latency (obtained by using the `dsstat`(1M) and `iostat` commands)
- Drain rate of the queue, depending on latency
- Existing high-water mark
- Existing queue settings

The following procedure is an example showing how to set the queue.

1. **Set the desired queue size in FBAs.**

   Convert the data into 512-byte chunks.

   ```
   # sndradm -F new_FBA_size -W new_write_size [set-name]
   ```

2. **Check the queue size.**

```
# sndradm -P
autosync: on, max q writes: 4194304, max q fbas: 16384, mode: async
```

3. **Compute the size of the queue.**

   Refer to "Examples" on page 21.

Examples

Consider the following examples when the application is writing to one volume.

- For a link with 50 millisecond latency
  - For 32 Kbyte writes, the transfer rate would be 640 Kbytes per second network throughput, computed as (1/0.050) X 32 Kbytes
  - For 8 Kbyte writes, the transfer rate would be 160 Kbytes per second, computed as (1/0.050) X 8 Kbytes

The following two scenarios are based on these calculations.

Example 1

If the application performs 20 IOPs (8 Kbyte single-threaded writes) to a volume, we are filling the asynchronous queue by 20 x 8 Kbytes, or 160 Kbytes per second. Since the drain rate is greater or equal to the fill rate, the queue is not filled up completely.

Example 2

If the application performs 50 IOPs (8 Kbyte single-threaded writes) to a volume, we are filling the asynchronous queue by 50 x 8 Kbytes, or 400K per second (with a drain rate of 160K per second). The queue would be filled up in approximately 33 seconds (400 Kbytes minus 160 Kbytes, or 240 Kbytes to fill the 8 Mbyte queue), as the fill rate is much higher than the drain rate. By changing the queue size to 16 Mbytes, you are delaying the filling of the queue. However, if the write operation continues at the same pace, the queue eventually becomes full.

Summary

- If the fill rate is less than or equal to the drain rate, the default queue size is acceptable.
- If drain rate is less than the fill rate, increasing the queue size might provide a temporary solution. If the write operations continue for a prolonged period, the queue eventually fills.

APPENDIX **B**

Network Tuning

Note – Perform the procedures in this appendix on the primary and secondary remote mirror hosts.

TCP Buffer Size

The TCP buffer size is the number of bytes that the transfer control protocol allows to be transferred before it waits for an acknowledgement.

To get maximum throughput, it is critical to use optimal TCP send and receive socket buffer sizes for the link you are using. If the buffers are too small, the TCP congestion window will never fully open. If the receiver buffers are too large, TCP flow control breaks and the sender can overrun the receiver, causing the TCP window to shut down. This event is likely to happen if the sending host is faster than the receiving host. Overly large windows on the sending side are not a problem as long as you have excess memory.

The optimal buffer size is twice the bandwidth*delay product of the link:

buffer size = 2 * bandwidth * delay

For example, if your ping time is 50 milliseconds and the end-to-end network consists of all 100 Base-T Ethernet and OC3 (155 Mbps) links, the TCP buffers should be:

.05 seconds * (100 Mbits per 8 bits) = 625 Kbytes

Note – Increasing the buffer size to a much higher value over a shared network might impact the network performance. See the Solaris™ System Administrator Collection for information about tuning the size.

TABLE B-1 shows the maximum possible throughput for a 100Base-T network.

TABLE B-1 Network Throughput and Buffer Size

Latency	Buffer Size = 24 Kbytes	Buffer Size = 256 Kbytes
10 milliseconds	18.75 Mbits per second	100 Mbits per second
20 milliseconds	9.38 Mbits per second	100 Mbits per second
50 milliseconds	3.75 Mbits per second	40 Mbits per second
100 milliseconds	1.88 Mbits per second	20 Mbits per second
200 milliseconds	0.94 Mbits per second	10 Mbits per second

Viewing and Tuning TCP Buffer Sizes

You can view and tune your TCP buffer size by using the `/usr/bin/netstat`(1M) and `/usr/sbin/ndd`(1M) commands. TCP parameters to consider tuning include:

- `tcp_max_buf`
- `tcp_cwnd_max`
- `tcp_xmit_hiwat`
- `tcp_recv_hiwat`

Consider the following:

- Restart the remote mirror software by rebooting, allowing the software to use the new buffer size
- However, after you shut down and restart your server, the TCP buffers return to a default size

The solution is to set the values in a startup script as described in "Example: To Set and Verify the Buffer Size in a Startup Script" on page 26.

▼ To View TCP Buffers and Values

1. **View all TCP buffers by typing:**

```
# /usr/sbin/ndd /dev/tcp ? | more
```

2. **View individual settings by specifying the buffer name; the following example shows a value of 1073741824:**

```
# /usr/sbin/ndd /dev/tcp tcp_max_buf
1073741824
```

▼ To View Buffer Sizes for a Socket

- **Use the** `/usr/bin/netstat` **(1M) command to view the buffer size for a particular network socket.**

  For example, view the size for port 121, the default remote mirror port:

```
# netstat -na |grep 121
*.121                *.*                0      0 262144      0 LISTEN
192.168.112.2.1009   192.168.111.2.121   263536     0 263536     0 ESTABLISHED
192.168.112.2.121    192.168.111.2.1008  263536     0 263536     0 ESTABLISHED
# netstat -na |grep rdc
*.rdc                *.*                0      0 262144      0 LISTEN
ip229.1009           ip230.rdc          263536     0 263536     0 ESTABLISHED
ip229.rdc            ip230.ufsd         263536     0 263536     0 ESTABLISHED
```

The value 263536 shown in this example is the 256 Kbyte buffer size. It must be set identically in the primary and secondary hosts.

▼ Example: To Set and Verify the Buffer Size in a Startup Script

---

Note – Create this script on the primary and secondary hosts.

---

1. **Create the script file in a text editor using the following values:**

```
#!/bin/sh
   ndd -set /dev/tcp tcp_max_buf 16777216
   ndd -set /dev/tcp tcp_cwnd_max 16777216
# increase DEFAULT tcp window size
   ndd -set /dev/tcp tcp_xmit_hiwat 262144
   ndd -set /dev/tcp tcp_recv_hiwat 262144
```

2. **Save the file as** `/etc/rc2.d/S68ndd` **and exit the file.**
3. **Shut down and restart your server.**
4. **Verify the size as shown in** **"To View Buffer Sizes for a Socket" on page 25.**

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 USA
650-960-1300 Fax 650-969-9131

For U.S. sales office locations, call: 800-821-4643

In other countries, call:

Corporate headquarters:	650-960-1300
Intercontinental sales:	650-688-9000