

# Database Snapshot Performance Considerations under I/O-Intensive Workloads

SQL Server Best Practices Article

Writers: Sanjay Mishra, Microsoft Corporation

Ron Talmage, Solid Quality Learning

Technical Reviewers: Sunil Agarwal, Peter Byrne, Mark Souza, Prem Mehra, Mike Ruthruff, Burzin Patel, Michael Thomassy

Published: February 2008

Applies To: SQL Server 2005

### Copyright

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

 2008 Microsoft Corporation. All rights reserved.

Microsoft, SQL Server, Windows, and Windows Serverare either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

### Table of Contents

[Introduction 1](#_Toc189760093)

[Overview 1](#_Toc189760094)

[Database Snapshot Creation Steps 1](#_Toc189760095)

[Test Objectives 4](#_Toc189760096)

[Test Methodology 4](#_Toc189760097)

[Test Hardware and Software 4](#_Toc189760098)

[Test Results and Observations 5](#_Toc189760099)

[How Much Time Does It Take to Create a Database Snapshot? 5](#_Toc189760100)

[Transaction and I/O Patterns during Database Snapshot Creation 7](#_Toc189760101)

[Transaction Throughput and Response Time with Multiple Database Snapshots 9](#_Toc189760102)

[Impact of Database Snapshot on Index Creation 10](#_Toc189760103)

[Database Snapshot Data is Current as of the Beginning of the Analysis Phase 12](#_Toc189760104)

[Restoring the Source Database from a Database Snapshot 13](#_Toc189760105)

[Recommendations 14](#_Toc189760106)

[Conclusion 15](#_Toc189760107)

[Appendix A: Test Environment 16](#_Toc189760108)

#### Introduction

Database snapshots in SQL Server 2005 Enterprise Edition give you a consistent view of a database as of a certain point in time. The database snapshot appears to you as a read-only static database. You can create multiple database snapshots on a source database. You can revert the source database to an earlier state by restoring it from a database snapshot.

Database snapshot performance is closely tied to the I/O capabilities of the system. This white paper summarizes the results of testing database snapshot performance under intense I/O activity on a large database. We measured the impact of creating a single database snapshot and multiple database snapshots under varying transaction loads as well as the performance effect of database snapshots on index creation and rebuild. This white paper provides some best practice recommendations for using database snapshots under I/O-intensive conditions.

#### Overview

Database snapshots use a copy-on-first-write technique for each database page that is changed for the first time after the database snapshot is created. When a database page is changed the first time after the database snapshot is created, the original page, as it existed before the change, is synchronously written to the database snapshot sparse file. After an original page is written to a database snapshot sparse file, any subsequent changes to that page in the source database do not need to be written to that database snapshot again. If a given database page is not changed after the database snapshot was created, queries to the database snapshot that need data from the given page read the unchanged data from the source database. If a page changes after the database snapshot was created, that page is read from the database snapshot file.

For detailed information on how database snapshots work, see [Database Snapshots](http://msdn2.microsoft.com/en-us/library/ms175158.aspx) in SQL Server Books Online.

Queries to a database snapshot read from the source database data files or the database snapshot files. SQL Server must also write the originals of newly changed pages to the database snapshot file. Therefore, as the rate of updates to previously unchanged pages in the source database increases, the I/O activity of the database snapshot also increases. If database I/O write activity is spread across the source database, there is a correspondingly high rate of I/O writes to the database snapshot.

The purpose of our testing was to determine how database snapshot behavior changes when I/O rates for both read and write to a database increase.

##### Database Snapshot Creation Steps

Before exploring the performance behavior of a database snapshot, it is important to understand what happens when a database snapshot is created. To understand the internal events that take place when a database snapshot is created, the trace flags listed in Table 1 can be used.

|  |  |
| --- | --- |
| **Trace Flag** | **Description** |
| 3605 | Causes the output of other trace flags to be written to the ERRORLOG |
| 3504 | Prints the checkpoint flush timings |
| 3400 | Prints the recovery timings |
| 1810 | Prints the file create/open/close timings |

Table 1: Trace flags used for displaying steps during database snapshot creation

**Note**   These trace flags are used only for the purpose of listing various events that take place during database snapshot creation. It is strongly recommended that these trace flags should not be set during normal operations or on a production system, otherwise the ERRORLOG will be flooded with messages.

Using the trace flags listed in Table 1, the ERRORLOG shows messages about the events taking place during database snapshot creation. These are listed in Table 2.

|  |  |
| --- | --- |
| **Database Snapshot Creation Steps** | **Observation in the ERRORLOG** |
| CREATE DATABASE … AS SNAPSHOT command issued | Database snapshot files created on disk. You will see file create and close messages in the ERRORLOG. |
| Checkpoint occurs | SQL Server issues checkpoints at the beginning of database snapshot creation. You will see FlushCache messages in the ERRORLOG. |
| Database snapshot files opened | Database snapshot file open messages in ERRORLOG. |
| Database snapshot startup | Starting up database (the database snapshot) message in ERRORLOG. |
| Recovery starts | Recovery starting message in ERRORLOG. |
| Analysis phase | Analysis start and progress messages in ERRORLOG.  Analysis complete message. |
| Redo phase | Redo start and progress messages in ERRORLOG. |
| Undo phase | Undo start and progress messages in ERRORLOG. |
| Recovery finished | Recovery complete message in ERRORLOG. |
|  | Normal checkpointing resumes. You will see FlushCache messages in the ERRORLOG. |

Table 2: Database snapshot creation steps as observed using the selected trace flags

These events are discussed in detail in the following paragraphs:

1. Database snapshot data files are created on disk. A database snapshot is a separate database from the source database. A database snapshot has its own data files—exactly the same number as the source database but no transaction log files. Each data file in the database snapshot corresponds to a data file in the source database. The first thing you will notice after you issue the command to create a database snapshot is that data files are created in the drives and folders you specified in the CREATE DATABASE … AS SNAPSHOT command. Unlike the source database data files, the data files of the database snapshot are sparse files. They do not contain any data when they are initially created. Therefore, creating the database snapshot files does not take much time. You will see information in the ERRORLOG similar to the following for each file that is created. Creating these files takes only a few seconds.

2006-10-23 21:48:07.22 spid1057 Creating file P:\MSSQL\_root.mdf. 2006-10-23 21:48:07.24 spid1057 Closing file (1):P:\MSSQL\_root.mdf.

1. SQL Server issues a checkpoint to flush all the pages in the cache to disk. You will see information in the ERRORLOG similar to the following:

2006-10-23 21:49:28.34 spid1057 FlushCache: cleaned up 732965 bufs with 507694 writes in 80907 ms (avoided 298164 new dirty bufs)

1. SQL Server opens the newly created data files of the database snapshot. You will see information in the ERRORLOG similar to the following:

2006-10-23 21:50:26.89 spid1057 Opening file (1):P:\MSSQL\_root.mdf.

1. SQL Server starts up the database snapshot. You will see information in the ERRORLOG similar to the following:

2006-10-23 21:56:05.46 spid1057 Starting up database 'snapshot\_test\_2'.

1. The database snapshot is recovered
2. Recovery starting message in ERRORLOG:

2006-10-23 21:56:05.46 spid1057 RECOVERY (snapshot\_test\_2, 7): Starting

1. Analysis start and progress messages in ERRORLOG:

2006-10-23 21:56:05.46 spid1057 RECOVERY (snapshot\_test\_2, 7): Analysis starting

2006-10-23 21:56:05.46 spid1057 Analysis of database 'snapshot\_test\_2' (7) is 0% complete (approximately 2719 seconds remain). This is an informational message only. No user action is required.

**Note**   Database snapshot data is current as of the start of the analysis phase; see [Database Snapshot Data is Current as of the Beginning of the Analysis Phase](#_Database_Snapshot_Data) later in this document.

1. Analysis complete message:

2006-10-23 21:56:36.81 spid1057 Analysis of database 'snapshot\_test\_2' (7) is 100% complete (approximately 0 seconds remain). This is an informational message only. No user action is required.

1. Redo start and progress messages in ERRORLOG:

2006-10-23 21:56:37.21 spid1057 RECOVERY (snapshot\_test\_2, 7): Redo starting

2006-10-23 21:56:37.21 spid1057 RECOVERY (snapshot\_test\_2, 7): Estimate REDO 815938560 bytes, UNDO 1560650 records. Rampup is OFF.

2006-10-23 21:56:37.21 spid1057 Recovery of database 'snapshot\_test\_2' (7) is 0% complete (approximately 86795 seconds remain). Phase 2 of 3. This is an informational message only. No user action is required.

1. Undo progress messages in ERRORLOG:

2006-10-23 22:16:34.06 spid1057 Recovery of database 'snapshot\_test\_2' (7) is 17% complete (approximately 5669 seconds remain). Phase 3 of 3. This is an informational message only. No user action is required.

1. Recovery complete message in ERRORLOG:

2006-10-23 22:16:59.80 spid1057 SQL recovery (analysis, redo and undo) completed for database (snapshot\_test\_2, 7)

1. Normal checkpointing resumes:

2006-10-23 22:17:10.44 spid1057 FlushCache: cleaned up 72141 bufs with 39990 writes in 10625 ms (avoided 426 new dirty bufs)

Understanding the internal processes of database snapshot creation helps you understand the performance behavior of database snapshots, which is the focus of the remainder of this white paper.

#### Test Objectives

The objective of our testing was to determine useful recommendations for database snapshots under high rates of I/O activity. We attempted to find answers to the following questions:

* How much time does it take to create a database snapshot?
* What impact does a database snapshot have on application transaction throughput and performance?
* What is the impact of multiple database snapshots on application transaction throughput and performance?
* How does creating a database snapshot affect index creation and index rebuild?
* What are the best disk configurations for database snapshots?
* How much time does it take to restore a database from a database snapshot?

#### Test Methodology

Our goal was to find results that would be useful for production applications. Accordingly, we built a system that would be commonly found in a production environment.

##### Test Hardware and Software

Our database server was a 4-socket dual-core 64-bit AMD system from HP, running Windows® Server® 2003 SP1 x64 with SQL Server™ 2005 x64 Enterprise Edition SP1.

The application was deliberately configured to issue inserts, updates, deletes, and reads in a relatively even distribution across a large database. This helped ensure that a high rate of disk I/O would be performed against the source database.

The database data, tempdb, and database snapshot files were all stored on an EMC CX-700 Storage Area Network (SAN). For details about the server as well as the storage used for storing the source database and the database snapshot data, see [Appendix A](#_Appendix_A:_Test).

Our final storage configuration was as follows:

|  |  |  |
| --- | --- | --- |
| **Drive** | **RAID Configuration** | **Description** |
| J: | 8 disks (4+4 RAID 10) | Database log files |
| K: | 8 disks (4+4 RAID 10) | Tempdb data files |
| L: | 8 disks (4+4 RAID 10) | Tempdb log file |
| P: | 28 disks (14+14 RAID 10) | Database snapshot files |
| Q: | 80 disks (40+40 RAID 10) | Database data files |
| X: | 8 drives RAID 0 (300 GB) | Backups |

Table 3: Storage configuration for database snapshot testing

#### Test Results and Observations

This section describes the results of our testing and some observations about the database snapshots under I/O-intensive loads.

##### How Much Time Does It Take to Create a Database Snapshot?

When there is no or very little write activity happening in the source database, creating a database snapshot is very quick. However, as the write activity on the source database increases, the time it takes to create a database snapshot also increases, as shown in Figure 1.

The workload we chose performs random writes on pages scattered throughout the 1‑terabyte database. Transaction volume is proportional to the number of users. So, in a way, the number of pages changed per second is proportional to the number of active user connections.

The more pages that are changed (for the first time) by the application while the database snapshot is being created, the more time it takes to create the database snapshot. If these user sessions are doing only reads and not writes, database snapshot creation time is much shorter, even when there is a high number of concurrent active users.

Figure 1: Database snapshot creation time with varying workload

Database snapshot creation is comprised of many steps within SQL Server, as explained earlier. Which of these steps takes longer as the user load (and the number of pages changed) increases? Figure 2 shows a breakdown of the timing of the different steps that take place during a database snapshot creation.

Figure 2: Timing breakdown of database snapshot creation steps

Most database snapshot creation time is spent in the recovery (startup, analysis, redo, and undo) phase. If many data changes are taking place during the creation of the database snapshot, it takes longer to complete the recovery and therefore, longer to create the database snapshot.

If you have already created one database snapshot, does it take the same amount of time to create another database snapshot on the same source database? A database snapshot adds I/O load to the system. All first-time changes (since the database snapshot was created) to the pages in the source database result in writes of the corresponding pages in the database snapshot. If the system is overloaded with I/O requests, creating another database snapshot may take longer than creating the previous snapshot. Figure 3 displays the time it took to create up to five database snapshots with different levels of workloads.

Figure 3: Amount of time to create up to five database snapshots under varying workloads

As indicated in Figure 3, it took longer to create each subsequent database snapshot than the time it took to create the previous database snapshot. This is because each database snapshot adds I/O load on the system. With each additional database snapshot, every first-time change to a page in the source database now must be written to one additional database snapshot.

##### Transaction and I/O Patterns during Database Snapshot Creation

As discussed in the previous section, creating a database snapshot adds extra load on the system, especially I/O load. Therefore, database snapshot creation may have an impact on the performance of the source database.

When a database snapshot is created under I/O-intensive conditions, transaction throughput is affected, as shown in the System Monitor graph (Figure 4) of a typical database snapshot creation:

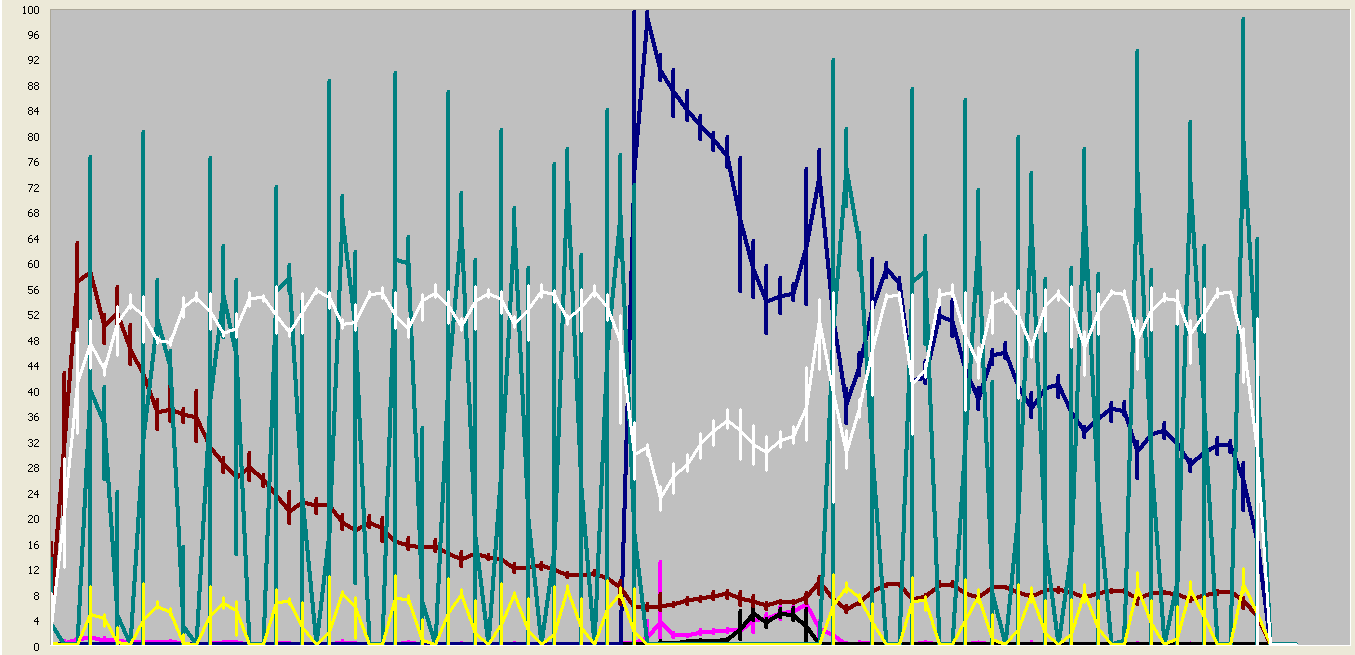


Figure 4: Transaction load pattern for a typical database snapshot creation under I/O-intensive conditions

Figure 4 shows a System Monitor graph from a typical database snapshot creation under moderately heavy I/O load, for a duration of 30 minutes. The graph lines are interpreted as follows:

|  |  |  |
| --- | --- | --- |
| **Line Color** | **Counter** | **Multiplier** |
| White | Transactions/sec in the source database | .1 |
| Yellow | Checkpoint Pages/sec | .001 |
| Green | Disk Write Bytes/sec to the source database data drive | .000001 |
| Blue | Disk Write Bytes/sec to the database snapshot data drive | .000001 |
| Black | Disk Read Bytes/sec from the database snapshot drive | .000001 |
| Pink | Disk Read Bytes/sec to the source database log drive | .000001 |

Table 4: Perfmon counters measured during the creation of a database snapshot

Transaction throughput (in white) varies slightly as the system stabilizes, lowering during checkpoints. After the initial checkpoint of the database snapshot creation, transaction throughput dips. After the database snapshot is created and regular checkpointing begins again, the transaction throughput slowly returns to normal.

##### Transaction Throughput and Response Time with Multiple Database Snapshots

Whenever a page is updated for the first time on the source database, the original copy of the page is synchronously written to the database snapshot file. This write activity impacts the throughput and response time of the application running on the source database. If you have more than one database snapshot created on a source database, every first‑time change to a page causes write activities on the data file of all the database snapshots, thereby causing a higher I/O load on the system. Therefore, the more database snapshots you have, the greater is the impact on the throughput and response time of a write-intensive workload.

The workload we used for testing issues random writes on pages scattered throughout the database. Therefore, the number of pages changed per second is approximately proportional to the number of active user sessions. We tested the application throughput with 0, 1, 3, and 5 pre-existing database snapshots for 100 and 500 concurrent active user sessions. The results are displayed in Figure 5.

Figure 5: Impact of multiple database snapshots on transaction throughput

On lower loads (with 100 active user sessions), the presence of multiple database snapshots does not have a significant impact on transaction throughput. However, with higher loads (500 active user sessions), transaction throughput gradually decreases as we add more database snapshots. As discussed earlier, with a higher number of active user connections, a large number of pages are updated, and each first-time page change causes the original page to be written to the database snapshot. Therefore, each additional database snapshot causes more work for each transaction, and hence transaction throughput decreases with the addition of each database snapshot.

The presence of database snapshots also impact the response time of the application. Figure 6 illustrates the response time of five different transactions with 0, 1, 3, and 5 database snapshots.

Figure 6: Impact of multiple database snapshots on transaction response time

Of the five transactions illustrated in Figure 6, Transaction 1 and Transaction 3 are the most write-intensive, and therefore have the most significant impact on response time as more database snapshots are added. Transaction 2 and Transaction 4 involve lower write activity, and therefore have less impact on response time as compared to Transactions 1 and 3. Transaction 5 is a read-only transaction and therefore, has no impact on response time with any number of database snapshots.

##### Impact of Database Snapshot on Index Creation

Creating or rebuilding an index is an I/O-intensive operation, especially for a clustered index. We compared the duration of creating a clustered and a nonclustered index on the largest table (364 million rows) in the database. We varied the following conditions:

* Workload and no workload
* Creating a clustered index and creating a nonclustered index
* One database snapshot versus no database snapshots

We measured the effect of database snapshot creation by first issuing the CREATE INDEX statement, then issuing the CREATE DATABASE SNAPSHOT statement. We used the ONLINE = ON option in order to sustain the workload during the test, and we also used the SORT\_IN\_TEMPDB = ON option to relieve pressure on the source database data file. Figure 7 illustrates the time required to create the index.

Figure 7: Impact of database snapshot creation on index creation

Creating a database snapshot while creating a large index has an impact on index creation time. The impact is more pronounced in the case of a clustered index or in the presence of a workload.

When you create or rebuild an index in the presence of one or more database snapshots, it takes longer to create the index. When rebuilding an index, even though there is no data change, rows are moved between pages, and therefore, pages are written to the database snapshot. Figure 8 illustrates the time it took to rebuild an existing clustered index on a large table in the presence of 0, 1, and 2 database snapshots.

Figure 8: Impact of existing database snapshots on index rebuild

As illustrated in Figure 8, it takes longer to rebuild an index in the presence of a database snapshot—the more database snapshots you have the longer it takes.

##### Database Snapshot Data is Current as of the Beginning of the Analysis Phase

One usage scenario of database snapshots is point-in-time reporting. Since a database snapshot provides a point-in-time image of the source database, it can be used to query the data as it existed at a certain point in time. It is important to note the exact point in time at which the database snapshot data is current. For systems with practical workloads, creating a database snapshot takes a noticeable amount of time; the data in the database snapshot is *not* current as of the time you issued the CREATE DATBASE … AS SNAPSHOT command, nor is it current as of the time the command completed. It is somewhere in the middle, as illustrated in Figure 9.

Time

CREATE DATABASE … AS SNAPSHOT command issued

Database Snapshot creation completed

Data in the DB snapshot is current as of this time

Figure 9: As of what point in time is the database snapshot current?

To determine the point at which database snapshot data is current, we created a small table and continuously inserted rows into it. This table has a column representing the timestamp of the insert. After the database snapshot was created, we queried the table from the database snapshot to find its most recent entry. The most recent row in the database snapshot corresponded to the start of the analysis phase of the database snapshot recovery, as reflected in the SQL Server ERORRLOG.

*Our conclusion is that data in the database snapshot is current as of the beginning of the analysis phase of the database snapshot recovery.* If the prior steps are time consuming, the database snapshot data will not be current as of the time you issued the CREATE DATABASE … AS SNAPSHOT command, but potentially a significant amount of time later. Data that is committed in the source database as of the beginning of the analysis phase of the database snapshot recovery will be present in the database snapshot. If there are long-running transactions in the source database prior to issuing the CREATE DATABASE … AS SNAPSHOT command, those transactions will not be present in the database snapshot.

##### Restoring the Source Database from a Database Snapshot

The source database can be reverted back to an earlier state by using the RESTORE DATABASE … FROM DATABASE\_SNAPSHOT command. Any data changes to the source database after the database snapshot was created will be lost. The revert operation rebuilds the log; therefore, you cannot apply the transaction log backups taken earlier.

Figure 10 illustrates the time it took to restore the source database from a database snapshot.

Figure 10: Time to restore from a database snapshot

As illustrated in Figure 10, the time it takes to restore from a database snapshot is nearly proportional to the size of the database snapshot. SQL Server stores database snapshot files as sparse files. Therefore, the **Size** value reported in Windows Explorer for the database snapshot files is the same as the database files of the source database, but that is not the actual size of the database snapshot sparse file. To find the actual size of the database snapshot files, look at the **Size on disk** value in the file properties. You can also find the actual size by querying the **size\_on\_disk\_bytes** column of **sys.dm\_io\_virtual\_file\_stats()**. For example:

SELECT DB\_NAME(database\_id), size\_on\_disk\_bytes

FROM sys.dm\_io\_virtual\_file\_stats(DB\_ID(N'AdventureWorks\_DBSS1'), 1)

#### Recommendations

Database snapshots are an I/O-intensive operation. If you are already operating with an I/O-heavy application, the following recommendations can help benefit overall system performance.

* Place the database, database log, and database snapshot files on separate independent disk volumes. Database snapshot creation involves high rates of reading from the source database data file(s) and from the source database transaction log, and writing to the database snapshot file(s). By keeping these on independent disks you lessen the risk of disk contention.
* Place the source database data files on the better-performing volume, which usually means volumes with the greater number of disk drives.
* Place the database snapshot files on good performing volumes as well.
* Try to minimize the number of database snapshots that are created successively under heavy I/O load, as subsequent creations do take longer when other database snapshots are already present.
* Avoid creating more than one database snapshot before applying user load, as the transaction throughput and response time is affected by pre-existing database snapshots.
* Use the I/O counters from System Monitor to determine that you have adequate I/O bandwidth to handle the expected load.
* Avoid creating database snapshots during index maintenance operations. Database snapshots have an impact on index creation or rebuild performance.
* Database snapshots can be an effective method of restoring a database to a previous point in time. The time required for restoring from a database snapshot is proportional to the size on disk of the database snapshot files.

#### Conclusion

Database snapshots can be a useful feature of SQL Server 2005 Enterprise Edition but they have important considerations for the I/O of your system. If your system is currently under I/O-heavy load, database snapshot creation may take longer than expected and have an impact on the transaction throughput of your system. By researching ahead, you can find ways to optimize their performance and minimize their impact.

**For more information, please visit:**

<http://technet.microsoft.com/en-us/sqlserver/bb671430.aspx>

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screenshots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screenshots, unclear writing?

Your feedback helps us improve the quality of white papers we release. [Send feedback](mailto:sqlbp@microsoft.com?subject=White%20Paper%20Feedback:%20Database%20Snapshot%20Performance%20Considerations%20under%20IO-Intensive%20Workloads).

#### Appendix A: Test Environment

Database server

|  |  |
| --- | --- |
| Operating System | Windows Server 2003 Enterprise x64 Edition SP1 |
| Manufacturer and Model | HP ProLiant DL585 G1 |
| Processors | Four dual-core AMD64 Family 15 Model 33 Stepping 0 AuthenticAMD ~2197 Mhz |
| Total Physical Memory | 32 GB |
| HBA | QLogic Fibre Channel Adapter |
| Network Adaptors | HP NC7782 Gigabit Server Adapter  HP NC7782 Gigabit Server Adapter #2 |
| SQL Server 2005 | SQL Server 2005 Enterprise Edition for x64 SP2 beta |

Application server

|  |  |
| --- | --- |
| Operating System | Windows Server 2003 Enterprise x64 Edition SP1 |
| Manufacturer and Model | Dell PowerEdge 6650 |
| Processors | Eight 32-bit single-core processors: x86 Family 15 Model 2 Stepping 5 GenuineIntel ~2786 Mhz |
| Total Physical Memory | 32 GB |
| Network Adaptors | Broadcom NetXtreme Gigabit Ethernet  Broadcom NetXtreme Gigabit Ethernet #2 |

Storage

|  |  |
| --- | --- |
| Manufacturer and Model | EMC Clariion CX-700 |
| Alignment Offset | 64 KB |