Transaction logging

by
Michael
Robinson

Level: Beginner
Works with: Domino 5.0
Updated: 08/02/99

Inside this article:

Related links:

Domino 5 Administration Help

NotesBench Web site

Domino Performance Zone

Get the PDF:

[Editor's note: We're excited to have this article on transaction logging, which we implemented on the Notes.net site. You'll learn about the performance benefits and more!]

Domino R5 takes a giant leap forward by now including *transaction logging*, an industry-standard technique and state-of-the-art solution for reliable data storage. With transaction logging enabled, the system captures database changes and writes them to the transaction log. Then if a system or media failure occurs, you can use the transaction log and a third-party backup utility to recover your database. You also get faster server restart times, greater database integrity, and much higher system availability. Finally, transaction logging also means the end of those long Fixup sessions!

One of the inherent benefits of using Domino transaction logging is an overall increase in system performance. In this article, we'll take a look at a performance analysis of Domino transaction logging on Windows NT. Before we get into the numbers, we'll start with a general overview of transaction logging and its benefits. We'll then describe the test methodology and test data, and finally summarize what the results mean to you.

Note: For in-depth information on using transaction logging, see the **Domino 5 Administration Help**.

An introduction to transaction logging: A better solution

Prior to R5, database-specific *transactions* (or operations) -- such as creating, modifying, or deleting documents; updating views; or changing database attributes -- required that the operation *commit to disk* before being considered successful. In other words, the transaction was either successful or failed; there was no middle ground. Even when you used database buffers (such as, the NOTES.INI setting NSF_BUFFER_POOL), the transaction and its data were still required to be committed to disk.

On an active server, the actual writing to disk could be a lengthy process. Modifications could occur on different parts of a database or across multiple databases. Then, the server's disk head had to move randomly over all areas of the disk to get to the proper track and sector for the data that was changed or updated. Repositioning the head in this "random" manner added to the total time required to complete a transaction. So, as the number of database users on an R4 server increased, so did the number of transactions as well as the average transaction completion time. Waiting for commits to complete (as opposed to returning from the operation and "trusting" that the system would eventually get the data to disk) was necessary so that in the event of a system failure, the Fixup task could fix the databases and restore them to a "clean" state.

Streamlining database transactions

Transaction logging in Domino R5 adds a dimension that streamlines all database transactions. With transaction logging, Domino posts transactions (or writes) to a series of log files *prior* to allowing any updates to the database. Successfully posting to the log is also considered a commit (the data is safely stored to persistent storage), allowing the database operation to

complete and continue. The transactions and their data are written to the actual database from the logs at some point in the future. In the event of a system failure, transaction log recovery can apply or undo only those transactions not written to disk at the time of the failure.

The biggest benefit of writing to the log *first*, then writing to the database is that all writes to log files are *sequential* in nature. Therefore, there is less head movement on the data disk(s), resulting in faster commits to disk. Why is there less head movement? Quite simply, random access is slower than sequential access. As mentioned above, random access requires the disk head to move randomly over the disk to find the proper data points. With sequential access, the disk head can move directly to the next available track on the disk.

System-level benefits
Transaction logging also has system-level benefits as well. At the operating system level, all file and disk operations are carried out by the OS kernel in either "kernel mode" or "privilege mode" (the process state where all file and disk operations are carried out by the OS kernel). This requires a context switch to the kernel thread to handle the task of moving data to and from disk. The longer the disk operations take, the more time the system spends waiting for I/O to complete and the *less* time the system spends doing the work of the application (that is, Domino).

Transaction logging improves the overall I/O throughput of Domino; therefore, it reduces kernel time and *increases* user time (the process state that all applications run). The reduction in time spent in kernel mode reduces the overall CPU cycles required to do the same work. Domino can then use those cycles. Transaction logging improves overall Domino server transaction throughput, which potentially allows the same server to support *more users.*

In short, using transaction logging:

- Streamlines data directory I/O
- Reduces kernel mode time and increases user mode time
- Increases user-perceived server responsiveness
- Increases server transaction throughput

Coupling this with the backup and archiving benefits of transaction logging, transaction logging makes your server run like it never ran before. (For more information on using transaction logging for backups, see the **Domino 5 Administration Help**.)

Setting up transaction logging

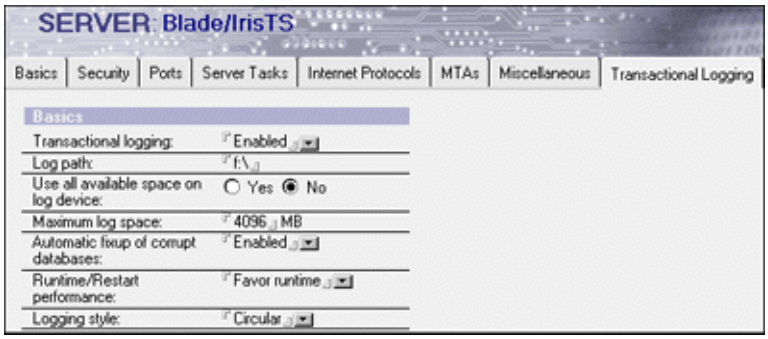
Transaction logging works with databases in Domino R5 format, but not with databases that use formats from earlier releases. After you enable transaction logging, all databases in R5 format are automatically logged.

To set up transaction logging on a Domino R5 server:

1. Allocate a space for the transaction logging directory. We recommend that you allocate a *separate* disk with at least 1GB of disk space for the transaction log.
2. Make sure that all databases to be logged reside in the Domino data directory, either at the root or in subdirectories.
3. Run Compact on your server (this will upgrade your databases to the new R5 format).
4. Open the Server document in the Domino Directory, click the Transactional Logging tab, and do the following:

   -- Choose "Enabled" for transaction logging.
   -- Enter the path name location of the transaction log (allocated in Step

1).
-- Enter the maximum log space in MB, between 192-4096MB (4GB). We recommend that you use 1028MB (1GB).
-- Choose "Enabled" for automatic fixup of corrupt databases.
-- Select "Favor runtime" in the Runtime/Restart performance field to set fewer recovery checkpoints in the transaction log, and improve runtime performance. (If you are concerned about server restart performance, select "Favor Restart" to record more recovery checkpoints, and improve restart performance.)
-- Select "Circular" for the logging style to continuously re-use the log files and overwrite the old transactions. (Use "Archiving" when you will be backing up the logs.)



5. Restart the Domino server.

Test methodology: Show me the numbers!

Now, let's look at the relative impact of turning on Domino transaction logging. Our test scenario used the new **NotesBench** R5 Mail workload on systems with transaction logging turned on and off. We compared the CPU utilization (both user and privileged mode utilization), average transactions per minute, and NotesBench response times. This section describes the system configurations, and more in-depth information about the workload.

System configurations

For our test scenario, we set up a Domino server with the following configuration:

- CPUs: Four Pentium II/400MHz with 1MB L2 Cache, IBM Netfinity 7000 M10
- Memory: 4GB RAM
- Three Ultra2 SCSI (80MB/s) controllers
- Six hardware arrays created across 12 disk drives
- Total of 22 9GB 10K RPM drives with the following configuration:
  --Physical C: One 9GB drive, RAID0, for OS
  --Physical D: One 9GB drive, RAID0, for page file
  --Physical E: One 9GB drive, RAID0, for Domino executables
  --Physical F: One 9GB drive, RAID0, for log files (**should** be its own drive!)
  --Logical G: Nine 9GB drives in EXP15 enclosure configured as RAID0, with four drives for user mail files, logs, and MAIL.BOX (total 36GB storage for primary \data directory on Channel 1 of SCSI Controller 1)
  --Logical M: Nine 9GB drives in EXP15 enclosure configured as RAID0, with four drives for user mail files, logs, and MAIL.BOX (total 36GB storage for secondary \data directory on Channel 1 of SCSI Controller 2)
- OS: Windows NT 4.0 Enterprise Edition with Service Pack 4
- Domino: Build 165 (R5 production build)

When setting up our test systems, our goal was to maximize the I/O bandwidth. Therefore, we distributed the file access as much as possible by setting up the OS, paging file, and Domino executables in separate *physical* drives. (It really isn't necessary to place these files on stripe sets.) In addition, we set up the log files on a *separate* drive. That way, we avoided having any other data written to the drive, and reduced any random head movement. Remember that writes to the log files are sequential, which is where we get our performance win. To further maximize performance, you can place the log file drive on its own channel, and on its own controller.

We placed the Domino data files (the Domino Directory, MAIL.BOX databases, and mail files) on two *separate* RAID0 stripe sets. Each stripe set is on a separate SCSI RAID controller. We placed 2500 mail files in both the primary and secondary data directories. This configuration allowed us to control which RAID set users would need to access, and isolate mail I/O traffic from application traffic.

In addition, we allocated 10 mailboxes on the Router/SMTP-Basics tab in the Server Configuration document, and specified the following NOTES.INI settings:

- SERVER_POOL_TASKS=100 (This specifies 100 worker threads -- the new R5 mechanism for actually carrying out the database work of all connected users.)
- SERVER_MAX_CONCURRENT_TRANS=1000
- NSF_BUFFER_POOL_SIZE=768000000 (or 768MB)

About the R5Mail workload
The R5Mail workload models an active user reading and sending mail, as well as scheduling an appointment, and sending meeting invitations. The script sends six messages (two memos, two appointments, and two invitations) to three recipients every 30 minutes. For each iteration of the 15-minute script, the client:

- Reads five documents
- Updates two documents
- Deletes one document
- Adds one document
- Scrolls down one view
- Opens and closes one database

For additional details on this workload, see the **User Profiles document** on the NotesBench Web site. (You must **register on the NotesBench site** in order to access this document.)

What did we find out?

With our test configuration, we found an across-the-board improvement in performance when turning on transaction logging. We compared the CPU utilization, user utilization, privilege utilization, average transactions per minute, and NotesBench response times.

The following chart shows the results of our R5Mail-only test for 5000 users. The test time took about six hours. Notice that with transaction logging turned on, the average transactions per minute increased by about 16 percent. The NotesBench response time was approximately 2.6 times faster. The reduction in privilege mode utilization means that more CPU time was dedicated to user mode execution. This means that Domino was able to complete more work during the testing time, which is directly reflected in the increased average-transactions-per-minute numbers. Also, since transaction logging streamlines writes to the Data drive, we also experienced a reduction in I/O utilization.

	% CPU util.	% User Mode util.	% Disk Time	% Priv. Mode util.	Avg. Trans/ Min	Response Time (ms)
Logging Off	31.00	22.45	100	9.34	7914	158
Logging On	25.24	20.28	87	4.96	9225	60

As you can see, simply by adding an additional disk for the log file directory and turning on transaction logging, you can realize an *immediate* performance improvement!

ABOUT THE AUTHOR

Michael Robinson is currently a Senior Software Engineer in the Database Performance Group. While at Iris, he has written performance stress and characterization tools for Domino Web Server, Calendar and Scheduling, and the Notes Database Engine. He has added several of those stress workloads to the official Lotus Notes Benchmark tool - NotesBench. He is also the developer of Lotus Server.Load (a free GUI Notes load generation tool), which is available on the Domino R5.0.1 CD and on the **Lotus Performance Zone Web site**. Previously, Michael spent a year working in Lotus Product Management as the Lotus Notes UNIX Product Manager. Prior to that, he spent four and a half years in Hewlett Packard's Workstation group writing HP-UX bootstrap firmware, and he spent time working as a Digital Designer Engineer. Michael is currently pursuing an M.S. in Engineering, and has a B.S. from the University of Miami, FL in Computer Engineering.