Microsoft TechNet

Storage Top 10 Best Practices

Published: October 17, 2006

Proper configuration of IO subsystems is critical to the optimal performance and operation of SQL Server systems. Below are some of the most common best practices that the SQL Server team recommends with respect to storage configuration for SQL Server.

1 Understand the IO characteristics of SQL Server and the specific IO requirements / characteristics of your application.

In order to be successful in designing and deploying storage for your SQL Server application, you need to have an understanding of your application's IO characteristics and a basic understanding of SQL Server IO patterns. Performance monitor is the best place to capture this information for an existing application. Some of the questions you should ask yourself here are:

- What is the read vs. write ratio of the application?
- What are the typical IO rates (IO per second, MB/s & size of the IOs)? Monitor the perfmon counters:
    1. Average read bytes/sec, average write bytes/sec
    2. Reads/sec, writes/sec
    3. Disk read bytes/sec, disk write bytes/sec
    4. Average disk sec/read, average disk sec/write
    5. Average disk queue length
- How much IO is sequential in nature, and how much IO is random in nature? Is this primarily an OLTP application or a Relational Data Warehouse application?

To understand the core characteristics of SQL Server IO, refer to SQL Server 2000 I/O Basics.

2 More / faster spindles are better for performance

- Ensure that you have an adequate number of spindles to support your IO requirements with an acceptable latency.
- Use filegroups for administration requirements such as backup / restore, partial database availability, etc.
- Use data files to "stripe" the database across your specific IO configuration (physical disks, LUNs, etc.).

3 Try not to "over" optimize the design of the storage; simpler designs generally offer good performance and more flexibility.

- Unless you understand the application very well avoid trying to over optimize the IO by selectively placing objects on separate spindles.
- Make sure to give thought to the growth strategy up front. As your data size grows, how will you manage growth of data files / LUNs / RAID groups? It is much better to design for this up front than to rebalance data files or LUN(s) later in a production deployment.

4 Validate configurations prior to deployment

- Do basic throughput testing of the IO subsystem prior to deploying SQL Server. Make sure these tests are able to achieve your IO requirements with an acceptable latency. SQLIO is one such tool which can be used for this. A document is included with the tool with basics of testing an IO subsystem. Download the SQLIO

Disk Subsystem Benchmark Tool.

- Understand that the of purpose running the SQLIO tests is not to simulate SQL Server's exact IO characteristics but rather to test maximum throughput achievable by the IO subsystem for common SQL Server IO types.
- IOMETER can be used as an alternative to SQLIO.

5 **Always place log files on RAID 1+0 (or RAID 1) disks. This provides:**

- better protection from hardware failure, and
- better write performance.

  Note: In general RAID 1+0 will provide better throughput for write-intensive applications. The amount of performance gained will vary based on the HW vendor's RAID implementations. Most common alternative to RAID 1+0 is RAID 5. Generally, RAID 1+0 provides better write performance than any other RAID level providing data protection, including RAID 5.

6 **Isolate log from data at the physical disk level**

- When this is not possible (e.g., consolidated SQL environments) consider I/O characteristics and group similar I/O characteristics (i.e. all logs) on common spindles.
- Combining heterogeneous workloads (workloads with very different IO and latency characteristics) can have negative effects on overall performance (e.g., placing Exchange and SQL data on the same physical spindles).

7 **Consider configuration of TEMPDB database**

- Make sure to move TEMPDB to adequate storage and pre-size after installing SQL Server.
- Performance may benefit if TEMPDB is placed on RAID 1+0 (dependent on TEMPDB usage).
- For the TEMPDB database, create 1 data file per CPU, as described in #8 below.

8 **Lining up the number of data files with CPU's has scalability advantages for allocation intensive workloads.**

- It is recommended to have .25 to 1 data files (per filegroup) for each CPU on the host server.
- This is especially true for TEMPDB where the recommendation is 1 data file per CPU.
- Dual core counts as 2 CPUs; logical procs (hyperthreading) do not.

9 **Don't overlook some of SQL Server basics**

- Data files should be of equal size – SQL Server uses a proportional fill algorithm that favors allocations in files with more free space.
- Pre-size data and log files.
- Do not rely on AUTOGROW, instead manage the growth of these files manually. You may leave AUTOGROW ON for safety reasons, but you should proactively manage the growth of the data files.

10 **Don't overlook storage configuration bases**

- Use up-to-date HBA drivers recommended by the storage vendor
- Utilize storage vendor specific drivers from the HBA manufactures website
- Tune HBA driver settings as needed for your IO volumes. In general driver specific settings should come from the storage vendor. However we have found that Queue Depth defaults are usually not deep enough to support SQL Server IO volumes.
- Ensure that the storage array firmware is up to the latest recommended level.

- Use multipath software to achieve balancing across HBA's and LUN's and ensure this is functioning properly
- Simplifies configuration & offers advantages for availability
- Microsoft Multipath I/O (MPIO): Vendors build Device Specific Modules (DSM) on top of Driver Development Kit provided by Microsoft.

Top of page