Performance Baseline

Server: <Server Name>

10 February 2011

Joe Bloggs

Bloggs Co

Table of Contents

[Introduction 3](#_Toc285114270)

[Executive Summary 3](#_Toc285114271)

[Server Specification 3](#_Toc285114272)

[Operating System 4](#_Toc285114273)

[Performance Collection 4](#_Toc285114274)

[Performance Analysis 5](#_Toc285114275)

[SQL Server 6](#_Toc285114276)

[Performance Collection 6](#_Toc285114277)

[Performance Analysis 16](#_Toc285114278)

# Introduction

This performance baseline, for server <server name>, was created between 7 and 10 February 2011. The work was carried out by Joe Bloggs ([joe@bloggs.com.au](mailto:joe@bloggs.com.au)).

The major uses for this document include, troubleshooting performance related issues and to facilitate capacity planning.

A performance baseline should be created or updated:

* following the initial production deployment of a system,
* before and after applying major updates, like application and operating system service packs, and
* when deploying additional workload to a system e.g. an additional database is deployed to a SQL server instance.

# Executive Summary

The 7 day average percentage processor time has increased remarkably from 30% to 90% (refer to figure 1). The increase in processor use can be attributed to the deployment of the ‘bad database’ on server <server name>. It is recommended that the ‘bad database’ be isolated and/or redesigned.

The number of deadlocks/sec has also increased to an undesired number of 10 over the past 3 months, from an acceptable average of 1 in the 6 months prior (refer to figure 3). The cause can again be attributed to the ‘bad database’. In particular, poor table design (none of the ‘bad database’ tables have a defined primary key) is the root cause for the increase in deadlocks.

% free space on the U: drive has decreased steadily by 3.8% over the past 12 months (refer to figure 2). The current available capacity is likely to be exhausted within the next 5 months. A request to archive data from ‘old database’ has been raised to increase the available storage.

# Server Specification

Table : Server Specification

|  |  |
| --- | --- |
| Item | Value |
| Operating System version | *Windows Server 2003 R2 SP2* |
| Operating System Architecture | *x64* |
| SQL Server version | *SQL Server 2005 Enterprise Edition* |
| SQL Server Architecture | *x64* |
| #Processors | *2* |
| #Processor cores | *8* |
| Memory | *16GB* |
| C: (System) | *DAS,RAID1,64GB* |
| D: (Application Binaries) | *DAS,RAID1,64GB* |
| E: (Paging file) | *DAS,RAID0,64GB* |
| F: (System Databases) | *SAN,RAID10,40GB* |
| G: (System Database Logs) | *SAN,RAID0,4GB* |
| H: (TempDB) | *SAN,RAID0,20GB* |
| I: (TempDB Logs) | *SAN,RAID10,10GB* |
| U: (User Databases) | *SAN,RAID10,200GB* |
| V: (User Database Logs) | *SAN,RAID10,20GB* |

# Operating System

## Performance Collection

Table 2: Platform Performance Counters

| Object | Counter | Description | Threshold  (desired value) | Last Sampled Value | 7 day avg. | 7 day avg. (peaks) |
| --- | --- | --- | --- | --- | --- | --- |
| Memory | Available Mbytes | Unused physical memory (not page file) | > 100 MB |  |  |  |
| Memory | Pages Input/sec | Reads from hard disk per second to resolve hard pages | < 10 |  |  |  |
| Memory | Pages/sec | Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory\\Pages Input/sec and Memory\\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files. | < 2500 |  |  |  |
| Memory | Free System Page Table Entries | Free System Page Table Entries is the number of page table entries not currently in used by the system. This counter displays the last observed value only; it is not an average. | >10,000  > 24,000  on boot  (< 5000 indicates an urgent problem) |  |  |  |
| Paging file | %Usage | Amount of Page File in use, which indicates the server is substituting disk space for memory | < 70% |  |  |  |
| Paging file | %Usage Peak | Highest %Usage metric since the last time the server was restarted | < 70% |  |  |  |
| Process (sqlservr) | %Processor Time | Percentage of processor time spent on SQL Server process threads | < 80% |  |  |  |
| Process (msmdsrv) | %Processor Time | Percentage of processor time spent on SSAS process threads. | < 80% |  |  |  |
| Processor | %Processor Time | % Processor Time is the percentage of elapsed time that the processor spends to execute a non-Idle thread | < 80% |  |  |  |
| Processor | % Privileged Time | % Privileged Time is the percentage of elapsed time that the process threads spent executing code in privileged mode | < 30% |  |  |  |
| System | Processor Queue  Length | Number of threads waiting for CPU cycles,  where < 12 per CPU is good/fair, < 8 is better,  < 4 is best. | < 4/CPU |  |  |  |
| System | Context Switches/sec | Number of execution contexts switched in  the last second, where >6000 is poor, <3000  is good, and <1500 is excellent. | < 4/CPU |  |  |  |
| LogicalDisk | % Free Space | % Free Space is the percentage of total usable space on the selected logical disk drive that was free. | Growth rate, determines the desired value |  |  |  |
| LogicalDisk | Avg. Disk Sec/Read | Avg. Disk sec/Read is the average time, in seconds, of a read of data from the disk.  > 25ms for more than 2 minutes or continual spikes above requires investigation. High performance systems e.g. SQL, Exchange, DC, etc needs <10ms. | < 10ms |  |  |  |
| LogicalDisk | Avg. Disk Sec/Write | Avg. Disk sec/Write is the average time, in seconds, of a write of data to the disk. > 25ms for more than 2 minutes or continual spikes above requires investigation. High performance systems e.g. SQL, Exchange, DC, etc needs <10ms. | < 10ms |  |  |  |
| Network Interface | Bytes Total/sec | Bytes Total/sec is the rate at which bytes are sent and received over each network adapter, including framing characters. Network Interface\Bytes Total/sec is a sum of Network Interface\Bytes Received/sec and Network Interface\Bytes Sent/sec. | NA |  |  |  |

## Performance Analysis

Figure nn: Baselines over time

<Insert (a) graph(s) for major counters here to indicate baseline changes over time>

Figure 1: 7-day avg. % Processor Time (Feb 2010 to Feb 2011)

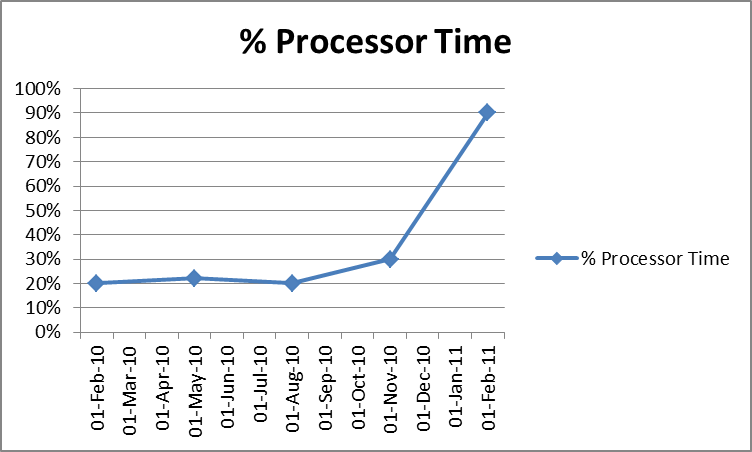
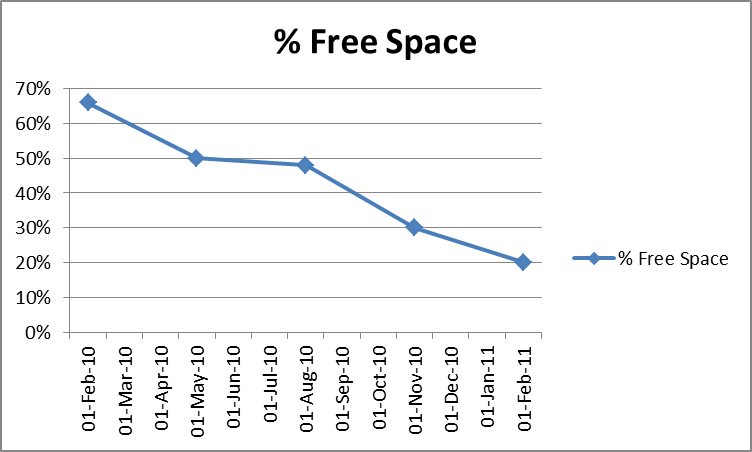


Figure 2: 7-day avg. % Free Space U: (Feb 2010 to Feb 2011)



# SQL Server

## Performance Collection

Table 3: SQL Server Performance Counters

| Object | Counter | Description | Threshold | Last Sampled Value | 7 day avg. | 7 day avg. (peaks) |
| --- | --- | --- | --- | --- | --- | --- |
| SQLServer:Access Methods | Forwarded Records/sec | Number of records fetched through forwarded record pointers. | < 10 per 100  Batch  Requests/Sec |  |  |  |
| SQLServer:Access Methods | Full Scans / sec | Number of unrestricted full scans. These can either be base table or full index scans.  High scan rates may be  caused by missing indexes, very small tables,  or requests for too many records. A sudden  increase in this value may indicate a statistics  threshold has been reached, resulting in an  index no longer being used. | Ignore unless high CPU coincides  with high scan rates |  |  |  |
| SQLServer:Access Methods | Index Searches/sec | Number of index searches. Index searches are used to start range scans, single index record fetches, and to reposition within an index. | 1 Full Scan/sec  per 1000 Index  Searches/sec |  |  |  |
| SQLServer:Access Methods | Page Splits/sec | Number of page splits per second that occur as a result of overflowing index pages. To avoid  page splits, review table and index design to  reduce non-sequential inserts or implement  fillfactor and pad\_index to leave more empty  space per page. NOTE: A high value for this  counter is not bad in situations where many  new pages are being created, since it includes  new page allocations. | < 20 per 100  Batch  Requests/Sec |  |  |  |
| SQLServer:Access Methods | Workfiles Created/sec | Number of work files created per second. For example, work files could be used to store temporary results for hash joins and hash aggregates. High values can indicate thrash in  tempdb and poorly coded queries. | < 20 |  |  |  |
| SQLServer:Access Methods | Worktables Created/sec | Number of work tables created per second. For example, work tables could be used to store temporary results for query spool, LOB variables, XML variables, and cursors. | < 20 |  |  |  |
| SQLServer:Access Methods | Worktables From Cache Ratio | Percentage of work tables created where the initial two pages of the work table were not allocated but were immediately available from the work table cache. A value  less than 90% may indicate insufficient  memory, since execution plans are being  dropped, or may indicate, on 32-bit  systems, the need for an upgrade to a  64-bit system. | > 90% |  |  |  |
| SQLServer:Access Methods | Table Lock Escalations/sec | The number of times locks on a table were escalated from page- or row-level to table-level. This number should, generally, be low. Frequent or even occasional spiking in this value may indicate poorly coded transactions. | Low |  |  |  |
| SQLServer:Transactions | Longest Running Transaction Time | The longest running time of any transaction in seconds. When blocking is high, check this counter to see if transactions are open for long periods of time. | NA |  |  |  |
| SQL Server:Memory Manager | Granted Workspace Memory (KB) | Total amount of memory granted to executing processes. This memory is used for hash, sort and create index operations. | NA |  |  |  |
| SQL Server:Memory Manager | Maximum Workspace Memory (KB) | Total amount of memory granted to executing processes. This memory is used primarily for hash, sort and create index operations. | NA |  |  |  |
| SQL Server:Memory Manager | Memory Grants Outstanding | Current number of processes that have successfully acquired a workspace memory grant | NA |  |  |  |
| SQL Server:Memory Manager | Memory Grants Pending | Current number of processes waiting for a workspace memory grant | <= 0 |  |  |  |
| SQL Server:Memory Manager | Total Server Memory (KB) | Total amount of dynamic memory the server is currently consuming. This value should grow until it is equal to Target Server Memory, as it populates its caches and loads pages into memory.  When it has finished, SQL Server is said to be in a “steady-state”. Until it is in steady-state, performance may be slow and IO may be higher. | NA |  |  |  |
| SQL Server:Memory Manager | Target Server Memory (KB) | Total amount of dynamic memory the server is willing to consume | NA |  |  |  |
| SQL Server:Databases | Data File(s) Size (KB) | The cumulative size of all the data files in the database. (\_Total or per database). Monitoring this counter is useful, for example, for determining the correct size of tempdb. | NA |  |  |  |
| SQL Server:Databases | Log Bytes Flushed/sec | Total number of log bytes flushed. Useful for determining trends and utilization of  the transaction log |  | NA |  |  |
| SQL Server:Databases | Log File(s) Size (KB) | The cumulative size of all the log files in the database. Useful for determining trends and utilization of the transaction log. | NA |  |  |  |
| SQL Server:Databases | Log File(s) Used Size (KB) | The cumulative used size of all the log files in the database. | NA |  |  |  |
| SQL Server:Databases | Log Flush Wait Time | Total wait time (milliseconds). | ~0 |  |  |  |
| SQL Server:Databases | Log Flush Waits/sec | Number of commits waiting on log flush. Effectively, the number of times per second that  SQL Server must wait for pages to be written to the transaction log. | ~0 |  |  |  |
| SQL Server:Databases | Log Flushes/sec | Number of log flushes. | NA |  |  |  |
| SQL Server:Databases | Log Growths | Total number of log growths for this database. Each time the transaction log grows, all user activity must halt until the log growth completes. Therefore, you want log growths to occur during predefined maintenance windows rather than during general working hours. | ~0 |  |  |  |
| SQL Server:Databases | Log Shrinks | Total number of log shrinks for this database. | ~0 |  |  |  |
| SQL Server:Databases | Log Truncations | Total number of log truncations for this database. Truncations should happen during log backups  or, on databases in simple recovery mode, at  checkpoint or the time period specified by  recovery interval. | NA |  |  |  |
| SQL Server:Databases | Percent Log Used | The percent of space in the log that is in use. Since all work in an OLTP database stops until writes can occur to the transaction log, it’s a very good idea to ensure that the log never fills completely. Hence, the recommendation to keep the log under 80% full. | < 80% |  |  |  |
| SQL Server:Buffer Manager | Free List Stalls/sec | Number of requests that had to wait for a free page. Any value above 2 means SQL Server needs more memory. | < 2 |  |  |  |
| SQL Server:Buffer Manager | Lazy Writes/Sec | Number of buffers written by buffer manager's lazy writer. Lower is better with zero being ideal. When greater than 20, this counter indicates a need for more memory. | < 20 |  |  |  |
| SQL Server:Buffer Manager | Checkpoint Pages/sec | Number of pages flushed by checkpoint or other operations that require all dirty pages to be flushed. Checkpoint frequency  is influenced by the recovery interval setting in sp\_configure. High values for this counter may indicate insufficient memory or that the recovery interval is too high. |  |  |  |  |
| SQL Server:Buffer Manager | Page Life Expectancy | Number of seconds a page will stay in the buffer pool without references. When under 300, this may indicate poor index design (leading to increased disk I/O and less effective use of memory) or, simply, a potential shortage of memory. | > 300 |  |  |  |
| SQL Server:Buffer Manager | Page Lookups/sec | Number of requests to find a page in the buffer pool. When the ratio of batch requests to page lookups crests 100, you may have inefficient execution plans or too many adhoc queries. | (Page lookups/  sec) / (Batch  Requests/  sec) < 100 |  |  |  |
| SQL Server:Buffer Manager | Page Reads/sec | Number of physical database page reads issued. Higher than 90 may be a yellow flag for poor indexing or insufficient memory. | < 90 |  |  |  |
| SQL Server:Buffer Manager | Page Writes/sec | Number of physical database page writes issued. Values over 90 should be cross checked with “lazy writer/sec” and “checkpoint” counters. If the other counters are also high, then it may indicate insufficient memory. | < 90 |  |  |  |
| SQL Server:Buffer Manager | Readahead/sec | Number of pages read in anticipation of use. If this value is makes up  even a sizeable minority of total Page Reads/sec  (say, greater than 20% of total page reads), you may have too many physical reads occurring. | < 20% of  Page Reads/  sec |  |  |  |
| SQL Server:Buffer Manager | Database Pages | Number of pages in the buffer pool with database content. | NA |  |  |  |
| SQL Server:Buffer Manager | Target Pages | Ideal number of pages in the buffer pool according the maximum memory granted to SQL Server in sp\_configure. | NA |  |  |  |
| SQL Server:Buffer Manager | Free Pages | Total number of pages on all free lists. A value less than 640 (5MB) indicates physical memory pressure. | > 640 |  |  |  |
| SQL Server:Buffer Manager | Stolen Pages/sec | Number of pages used for miscellaneous server purposes (including procedure cache). | NA |  |  |  |
| SQL Server:Buffer Manager | Buffer cache hit ratio | Percentage of pages that were found in the buffer pool without having to incur a read from disk. | Close to 100% |  |  |  |
| SQL Server:SQL Statistics | Batch requests/sec | Number of SQL batch requests received by server. A good general indicator for the activity level of the SQL Server. This counter is highly dependent on the hardware and quality of code running on the server. The more powerful the hardware, the higher this number can be, even on poorly coded applications. A value of 1000 batch requests/sec is easily attainable though a typical 100Mbs NIC can only handle about 3000 batch requests/sec. Many other counter thresholds depend upon batch requests/sec while, in some cases, a low (or high) number does not indicate poor processing power. You should frequently use this counter in combination with other counters, such as processor utilization or user connections. In version 2000, “Transactions/  sec” was the counter most often used to measure  overall activity, while versions 2005 and later use  “Batch Requests/sec”. Versions 2005 prior to SP2 measure this counter differently and may lead to some misunderstandings. | NA |  |  |  |
| SQL Server:SQL Statistics | SQL Attention Rate/sec | Number of attentions per second e.g. cancels and query timeouts per second. This number should be as low as possible. A high sustained number indicates frequent query timeout or end-user cancellation of queries. | ~0 |  |  |  |
| SQL Server:SQL Statistics | SQL Re-Compilations/sec | Number of SQL re-compiles. Number of times, per second, that Transact-SQL objects attempted to be executed but had to be recompiled before completion. This number should be at or near zero, since recompiles can cause deadlocks and exclusive compile locks.  This counter’s value should follow in proportion to “Batch Requests/sec” and “SQL Compilations/sec”. This needs to be 0 in your system as much as possible. | < 10% of the  number of  SQL Compilations/  sec |  |  |  |
| SQL Server:SQL Statistics | SQL Compilations/sec | Number of SQL compilations. Number of times that Transact-SQL compilations occurred, per second (including recompiles).  The lower this value is the better. High values often indicate excessive adhoc querying and should be as low as possible. If excessive adhoc querying is happening, try rewriting the queries as procedures or invoke the queries using sp\_executeSQL. When rewriting isn’t possible, consider using a plan guide or setting the database to parameterization forced mode. | < 10% of the  number of  Batch Requests/  Sec |  |  |  |
| SQL Server:Cursor Manager by Type | Active Cursors | Number of active cursors. Monitor cursor counters to see if there may be heavy use of server cursors since improper use can result in performance issues. | NA |  |  |  |
| SQL Server:SQL Errors | Errors/sec | Number of errors/sec. | ~0 |  |  |  |
| SQL Server:Deprecated Features | Usage | Feature usage since last SQL Server startup. Number of cancel and query timeouts/second or features used that are considered “deprecated”; i.e. features and commands Microsoft will not support in a future release. Run this counter when considering an upgrade to a newer version of SQL Server, update application accordingly. | ~0 |  |  |  |
| SQL Server:General Statistics | Logouts/sec | Total number of logouts started per second. Any value over 2 may indicate insufficient connection pooling. | < 2 |  |  |  |
| SQL Server:General Statistics | Logins/sec | Total number of logins started per second. Any value over 2 may indicate insufficient connection pooling. | < 2 |  |  |  |
| SQL Server:General Statistics | User Connections | Number of users connected to the system. This counter should roughly track with “Batch Requests/Sec”.  They should generally rise and fall together.  For example, blocking problems could  be revealed by rising user connections,  lock waits and lock wait time coupled  with declining batch requests/sec. | NA |  |  |  |
| SQL Server:Latches | Latch Waits/sec | Number of latch requests that could not be granted immediately and had to wait before being granted. Latches are lightweight means of holding a very transient server resource, such as an address in memory. | (Total Latch  Wait Time) /  (Latch Waits/  Sec) < 10 |  |  |  |
| SQL Server:Latches | Avg Latch Wait Time (ms) | Average latch wait time (ms) for latch requests that had to wait. | Correlate  to “Latch Waits/sec” and move up or down  with it accordingly |  |  |  |
| SQL Server:Latches | Total Latch Wait Time (ms) | Total latch wait time (milliseconds) for latch requests that had to wait in the last second. This value should stay stable compared to the number of latch waits/second. | (Total Latch  Wait Time) /  (Latch Waits/  Sec) < 10 |  |  |  |
| SQL Server:Locks | Lock Wait Time (ms) | Total wait time (ms) for locks in the last second. The total time spent waiting across all  transactions to acquire  a lock in the last second. Because SQL Server records a lock at the end of a locking event, remember that an application with huge transactions may have inflated lock wait times while still performing as expected. For example, an application that issues multi-million record updates might have very long lock wait times while performing exactly as it was designed. | NA |  |  |  |
| SQL Server:Locks | Avg Wait Time (ms) | The average amount of wait time (ms) for each lock request that resulted in a wait. An average wait time longer than 500ms may indicate excessive blocking. This value should generally correlate to “Lock Waits/sec” and move up or down with it accordingly. | < 500 |  |  |  |
| SQL Server:Locks | Lock Requests/sec | Number of new locks and lock conversions requested from the lock manager. This metric’s value should generally correspond to “Batch Requests/sec”. Values > 1000 may indicate queries are accessing very large numbers of rows and may benefit from tuning. | < 1000 |  |  |  |
| SQL Server:Locks | Lock Waits/sec | Number of lock requests that could not be satisfied immediately and required the caller to wait before being granted the lock. How many times users waited to acquire a lock over the past second. Values greater  than zero indicate at least some blocking is occurring, while a value of zero can quickly eliminate blocking as a potential root-cause problem. As with “Lock Wait  Time”, lock waits are not recorded by Perfmon until after the lock event completes. | 0 |  |  |  |
| SQL Server:Locks | Number of Deadlocks/sec | Number of lock requests that resulted in a deadlock. Since only a COMMIT, ROLLBACK, or deadlock can terminate a transaction (excluding failures or errors), this is an important value to track. Excessive deadlocking indicates a table or index design error or bad application design. | < 1 |  |  |  |
| SQL Server:Locks | Lock Timeouts/sec | Number of lock requests that timed out. This includes requests for NOWAIT locks.  A value greater than zero might indicate that user queries are not completing. The lower this value is the better. | < 1 |  |  |  |

## Performance Analysis

Figure nn: Baselines over time

<Insert (a) graph(s) for major counters to indicate baseline changes over time>

Figure 3: 7-day avg. number of Deadlocks/sec

