Example SQL Server Health Check Report

Prepared by
Colleen Morrow
David Maxwell
Michael Fal
Ben Miller

This is an example report with recommendations.
Contents
Executive Summary ......................................................................................................................... 3
Current Environment Overview ...................................................................................................... 4
   Server and Infrastructure ........................................................................................................... 4
   SQL Server Instance .................................................................................................................. 5
   Database Information ............................................................................................................... 5
SQL Server Performance Overview .............................................................................................. 6
   Host Performance (CPU and Memory) ...................................................................................... 6
   Disk and I/O Subsystem Performance ...................................................................................... 6
   SQL Instance Activity and Performance .................................................................................. 7
Query and Indexing Overview ...................................................................................................... 8
   Long Running and Expensive Queries ..................................................................................... 8
   Top Missing Indexes ............................................................................................................... 8
   Duplicate Indexes .................................................................................................................. 9
Analysis ........................................................................................................................................ 10
   Configuration ........................................................................................................................ 10
   Performance ........................................................................................................................... 11
   Disaster Recovery .................................................................................................................. 14
   Security .................................................................................................................................. 15
   Additional Concerns .............................................................................................................. 16
Conclusions .................................................................................................................................. 17
About the Authors ....................................................................................................................... 18
About UpSearch .......................................................................................................................... 22
Executive Summary

Scenario

SG Heavy Industries, LLC (“SG Heavy Industries” or “client”) engaged UpSearch, LLC (“UpSearch”) to facilitate an assessment of SG Heavy Industries’ current database platform in order to identify performance tuning and monitoring options.

Scope of Work

UpSearch provided SG Heavy Industries with Colleen Morrow, David Maxwell, Michael Fal and Ben Miller to gather and analyze operating system and SQL Server configurations and performance metrics, recommend performance changes based on SG Heavy Industries’ environment and industry best practices, and subsequently present this report.

Test Strategy

UpSearch ran a series of PowerShell scripts between February 15 and 16, 2016 (assessment period), which explored and gathered information from both the Windows operating system and from SQL Server on server ENTERPRISE. In addition, UpSearch ran a set of queries against the SQL Server instance and against the databases running on the server. UpSearch also collected performance counters for both the operating system and SQL Server.

Conclusion

UpSearch provided SG Heavy Industries this SQL Server Health Check Report after evaluating current configurations, performance metrics and existing maintenance procedures in order to better understand current performance bottlenecks. The findings and recommended actions of this report are not only intended to improve performance, but are also made with an eye towards simplifying and standardizing SQL Server management, ensuring business continuity and managing security.

The assessment revealed several configuration and performance related findings. The rough order of magnitude estimate for UpSearch’s time required to complete the ensuing recommendations is between 50 and 65 hours. See page 18 for time estimates by recommendation.

Next, UpSearch recommends;

1. SG Heavy Industries reviews UpSearch’s SQL Server Health Check report.
2. SG Heavy Industries and UpSearch reconvene to clarify findings, answer questions, consider short-term configuration and performance issues, as well as future roadmap options.
## Current Environment Overview

The following overview represents ENTERPRISE current state as of the *assessment period*.

### Server and Infrastructure

<table>
<thead>
<tr>
<th>Machine Name</th>
<th>ENTERPRISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan Date (UTC)</td>
<td>2/15/2016</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Dell</td>
</tr>
<tr>
<td>Operating System</td>
<td>Microsoft Windows Server 2008 R2 Datacenter</td>
</tr>
<tr>
<td>Version</td>
<td>6.1.7601</td>
</tr>
<tr>
<td>Service Pack</td>
<td>Service Pack 1</td>
</tr>
<tr>
<td>Install Date (UTC)</td>
<td>4/23/2014</td>
</tr>
<tr>
<td>Domain</td>
<td>STARFLEET.com</td>
</tr>
<tr>
<td>Role</td>
<td>Member Server</td>
</tr>
<tr>
<td>Physical Processors</td>
<td>1</td>
</tr>
<tr>
<td>Processor Cores</td>
<td>8</td>
</tr>
<tr>
<td>Logical Processors</td>
<td>16</td>
</tr>
<tr>
<td>Physical Memory (MB)</td>
<td>30720</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logical Disk Caption</th>
<th>Logical Disk Size (GB)</th>
<th>Logical Disk Free Space (GB)</th>
<th>Logical Disk Volume Name</th>
<th>Logical Disk Allocation Unit Size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td>60.00</td>
<td>18.63</td>
<td>OS</td>
<td>4,096</td>
</tr>
<tr>
<td>D:</td>
<td>239.99</td>
<td>21.29</td>
<td>Data</td>
<td>65,536</td>
</tr>
<tr>
<td>E:</td>
<td>600.00</td>
<td>57.01</td>
<td>Backups</td>
<td>4,096</td>
</tr>
<tr>
<td>Z:</td>
<td>152.57</td>
<td>90.48</td>
<td>Temp Storage</td>
<td>4,096</td>
</tr>
</tbody>
</table>
### SQL Server Instance

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>ENTERPRISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan Date (UTC)</td>
<td>2/15/2016</td>
</tr>
<tr>
<td>Install Date</td>
<td>4/10/2012</td>
</tr>
<tr>
<td>Startup Date</td>
<td>2/14/2016</td>
</tr>
<tr>
<td>Product Name</td>
<td>Microsoft SQL Server 2008 R2</td>
</tr>
<tr>
<td>Product Edition</td>
<td>Web Edition (64-bit)</td>
</tr>
<tr>
<td>Level</td>
<td>SP2</td>
</tr>
<tr>
<td>Platform</td>
<td>NT x64</td>
</tr>
<tr>
<td>Version</td>
<td>10.50.4339.0</td>
</tr>
<tr>
<td>Server Type</td>
<td>Standalone</td>
</tr>
<tr>
<td>Clustered</td>
<td>FALSE</td>
</tr>
<tr>
<td>Logical Processors</td>
<td>16</td>
</tr>
<tr>
<td>Total Server Memory (MB)</td>
<td>30720</td>
</tr>
<tr>
<td>SQL Server Memory In Use (MB)</td>
<td>26592</td>
</tr>
</tbody>
</table>

### Database Information

<table>
<thead>
<tr>
<th>Database Name</th>
<th>Owner</th>
<th>Compatibility Level</th>
<th>Recovery Model</th>
<th>Last Known DBCC Date</th>
<th>Last Full Backup</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>sa</td>
<td>SQL Server 2008 (100)</td>
<td>Simple</td>
<td>02/14/2016 1:00:08 AM</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>sa</td>
<td>SQL Server 2008 (100)</td>
<td>Full</td>
<td>02/14/2016 1:00:09 AM</td>
<td></td>
</tr>
<tr>
<td>msdb</td>
<td>sa</td>
<td>SQL Server 2008 (100)</td>
<td>Simple</td>
<td>02/14/2016 1:00:11 AM</td>
<td></td>
</tr>
<tr>
<td>PICARD_db</td>
<td>DEN_DATANOMIX\ckasten</td>
<td>SQL Server 2005 (90)</td>
<td>Full</td>
<td>07/27/2014 7:59:39 PM</td>
<td>02/07/2016 5:09:49 AM</td>
</tr>
<tr>
<td>tempdb</td>
<td>sa</td>
<td>SQL Server 2008 (100)</td>
<td>Simple</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database Name</th>
<th>Data File Count</th>
<th>Log File Count</th>
<th>Data File Size (MB)</th>
<th>Log File Size (MB)</th>
<th>Available Data Space (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>1</td>
<td>1</td>
<td>4.00</td>
<td>0.75</td>
<td>1.06</td>
</tr>
<tr>
<td>model</td>
<td>1</td>
<td>1</td>
<td>2.25</td>
<td>2.00</td>
<td>0.94</td>
</tr>
<tr>
<td>msdb</td>
<td>1</td>
<td>1</td>
<td>35.06</td>
<td>31.69</td>
<td>3.69</td>
</tr>
<tr>
<td>PICARD_db</td>
<td>1</td>
<td>1</td>
<td>201,608.31</td>
<td>22,250.50</td>
<td>7,451.81</td>
</tr>
<tr>
<td>tempdb</td>
<td>8</td>
<td>1</td>
<td>32,768.00</td>
<td>1.50</td>
<td>32,391.81</td>
</tr>
</tbody>
</table>
SQL Server Performance Overview

Host Performance (CPU and Memory)

Host performance indicates no overall stress from a server level. Overall low CPU usage (average 8.58%) indicates that the server is not under any CPU stress. Available memory averages at almost 2GB and does not fall below 1GB, showing that the server itself is not being starved for memory and has available resources for the operating system and other non-SQL Server functions. The low use of the paging file is evidence that the applications are not having to swap memory to disk, which is a key indicator of application memory pressure.

<table>
<thead>
<tr>
<th>Metric</th>
<th>MIN</th>
<th>AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paging File - Percent Usage</td>
<td>0.04</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td>Memory - Available MBBytes</td>
<td>1314.00</td>
<td>1958.20</td>
<td>2094.00</td>
</tr>
<tr>
<td>Processor - Percent Processor Time</td>
<td>0.03</td>
<td>8.58</td>
<td>99.99</td>
</tr>
</tbody>
</table>

Disk and I/O Subsystem Performance

The disk subsystem’s latency is within Microsoft’s recommendations for operation. Typically, it is expected that read operations should be less than 20 milliseconds and write operations be less than 10 milliseconds. The maximum values for these metrics indicate periodic spikes. For volumes hosting database files, this could be an indicator of specific queries that require tuning or a result of contention from the database log and data files being on the same volume.

<table>
<thead>
<tr>
<th>Metric</th>
<th>MIN</th>
<th>AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\ Avg. Disk ms/Read</td>
<td>0.00</td>
<td>2.19</td>
<td>421.48</td>
</tr>
<tr>
<td>D:(1) Avg. Disk ms/Read</td>
<td>0.00</td>
<td>1.88</td>
<td>302.22</td>
</tr>
<tr>
<td>D:(2) Avg. Disk ms/Read</td>
<td>0.00</td>
<td>1.94</td>
<td>335.48</td>
</tr>
<tr>
<td>D:(3) Avg. Disk ms/Read</td>
<td>0.00</td>
<td>1.79</td>
<td>308.60</td>
</tr>
<tr>
<td>D:(4) Avg. Disk ms/Read</td>
<td>0.00</td>
<td>1.80</td>
<td>274.49</td>
</tr>
<tr>
<td>E:\ Avg. Disk ms/Read</td>
<td>0.00</td>
<td>0.88</td>
<td>55.47</td>
</tr>
<tr>
<td>Z:\ Avg. Disk ms/Read</td>
<td>0.00</td>
<td>0.89</td>
<td>20.18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>MIN</th>
<th>AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\ Avg. Disk ms/Write</td>
<td>0.88</td>
<td>2.12</td>
<td>120.63</td>
</tr>
<tr>
<td>D:(1) Avg. Disk ms/Write</td>
<td>0.00</td>
<td>2.52</td>
<td>65.55</td>
</tr>
<tr>
<td>D:(2) Avg. Disk ms/Write</td>
<td>0.00</td>
<td>1.63</td>
<td>441.16</td>
</tr>
<tr>
<td>D:(3) Avg. Disk ms/Write</td>
<td>0.00</td>
<td>1.52</td>
<td>319.39</td>
</tr>
<tr>
<td>D:(4) Avg. Disk ms/Write</td>
<td>0.00</td>
<td>1.33</td>
<td>264.08</td>
</tr>
<tr>
<td>E:\ Avg. Disk ms/Write</td>
<td>0.00</td>
<td>1.82</td>
<td>304.30</td>
</tr>
<tr>
<td>Z:\ Avg. Disk ms/Write</td>
<td>0.00</td>
<td>1.47</td>
<td>101.03</td>
</tr>
</tbody>
</table>
SQL Instance Activity and Performance

In general, SQL Server is operating as expected in relation to internal processing. Query Compilations and Re-Compilations fall within expected ratios compared to the number of Batch Requests per second. The Page Life Expectancy counter is an overall indication of SQL Server memory efficiency. It is good to differentiate PLE for 7am-6pm on the server (business hours) versus the 24 hour average. In general, the trend for PLE is to average 4496 during the day, but when the server enters its low use timeframe, the PLE climbs steadily.

<table>
<thead>
<tr>
<th>Metric</th>
<th>MIN</th>
<th>AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Requests/sec</td>
<td>0.07</td>
<td>1109.25</td>
<td>6425.05</td>
</tr>
<tr>
<td>User Connections</td>
<td>32.00</td>
<td>82.67</td>
<td>206.00</td>
</tr>
<tr>
<td>Buffer cache hit ratio</td>
<td>93.30</td>
<td>99.98</td>
<td>100.00</td>
</tr>
<tr>
<td>Page life expectancy</td>
<td>953</td>
<td>15315</td>
<td>39506</td>
</tr>
<tr>
<td>Page life expectancy (Business Hours)</td>
<td>953</td>
<td>4496</td>
<td>39506</td>
</tr>
<tr>
<td>SQL Re-Compilations/sec</td>
<td>0.00</td>
<td>0.62</td>
<td>33.40</td>
</tr>
<tr>
<td>SQL Compilations/sec</td>
<td>0.00</td>
<td>99.23</td>
<td>676.51</td>
</tr>
</tbody>
</table>

SQL Server operates on a "waits and queues" system. This means that when tasks within the engine must "wait" for a resource to become available in order to complete, they are assigned a "queue" until that resource becomes available. These waits are tracked and can indicate what resources within the SQL Server instance are subject to performance issues.
Query and Indexing Overview

Long Running and Expensive Queries

SQL queries can perform poorly for a variety of reasons. Some factors are how they are written, inadequate hardware, or missing indexes to support the workload. It is important to identify the queries that are performing poorly based on:

- Number of Executions
- CPU cost
- Disk Impact (Reads and Writes)
- Duration

This is a listing of the most expensive queries currently running in the SG Heavy Industries environment, based on these factors.

<table>
<thead>
<tr>
<th>SQL Text</th>
<th>Execs</th>
<th>CPU</th>
<th>Reads</th>
<th>Writes</th>
<th>Duration(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{CURSOR} SELECT * FROM CUSTOMER_REG</td>
<td>295</td>
<td>2,223,620</td>
<td>2,130,748,982</td>
<td>2,222,518</td>
<td>2,454,634,124</td>
</tr>
<tr>
<td>`{CURSOR} SELECT * FROM CUSTOMER_REG_EXTENSION</td>
<td>73</td>
<td>498,423</td>
<td>615,087,877</td>
<td>990,457</td>
<td>549,260,223</td>
</tr>
<tr>
<td>`{CURSOR} SELECT PROCESS_OBJECT_HISTORY.PROCESS_EVENT_NAME AS RAW_EVENT, PROCESS_OBJECT_HISTORY...</td>
<td>10</td>
<td>71,416</td>
<td>4,170,635</td>
<td>3</td>
<td>96,526,517</td>
</tr>
<tr>
<td>`{CURSOR} SELECT SCRPT_DTA_OBJ_ID AS SDO, MAX(PARENT_TRAN_DTA_OBJ_ID) AS MAXPDO FROM TRAN_DTAOBJ_O...</td>
<td>1</td>
<td>13,182</td>
<td>290,561</td>
<td>9</td>
<td>30,535,488</td>
</tr>
<tr>
<td>`{CURSOR} SELECT * FROM (SELECT DISTINCT ROW_NUMBER() OVER (ORDER BY LNAME, FNAME ASC) AS ROWNUMBE...</td>
<td>4</td>
<td>1,218</td>
<td>243,913</td>
<td>100</td>
<td>22,423,025</td>
</tr>
<tr>
<td>`{CURSOR} SELECT * FROM PARTY</td>
<td>3</td>
<td>12,682</td>
<td>9,916,748</td>
<td>10,640</td>
<td>21,673,880</td>
</tr>
<tr>
<td>INSERT INTO &quot;PICARD_DB&quot;..&quot;CUSTOMER_SIGNER&quot; (&quot;SIGNER_ID&quot;, &quot;SIGNER_FIRST_NAME&quot;...</td>
<td>2</td>
<td>0</td>
<td>26</td>
<td>3</td>
<td>21,117,060</td>
</tr>
<tr>
<td>INSERT INTO EV_LOG (CASE_NUMBER, RECORD_TYPE, PARTY_ID, LAST_NAME, FIRST_NAME, MIDDLE_INITIAL, MA...</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>2</td>
<td>20,347,660</td>
</tr>
<tr>
<td>`{CURSOR} SELECT * FROM (SELECT DISTINCT ROW_NUMBER() OVER (ORDER BY LNAME, FNAME ASC) AS ROWNUMBE...</td>
<td>1</td>
<td>1,357</td>
<td>184,378</td>
<td>99</td>
<td>15,742,859</td>
</tr>
<tr>
<td>UPDATE &quot;PICARD_DB&quot;..&quot;CUSTOMER_SIGNER&quot; SET &quot;SIGNER_LINK&quot;=@P1 WHERE &quot;SIGNER_ID&quot;=@P2 AND &quot;SIGNER_DOCU...</td>
<td>2</td>
<td>0</td>
<td>28</td>
<td>3</td>
<td>12,778,330</td>
</tr>
</tbody>
</table>

Top Missing Indexes

SQL Server tracks individual query statistics as they are executed. From these statistics, it compiles a list of suggested indexes that could help query processing. Listed here is the top selection of these suggestions. It should be noted that implementation of these indexes should be reviewed and tested for impact, as SQL Server cannot take all factors of implementing these indexes into account.
Database.Schema.Table | Equality Columns | Included Columns
---|---|---
[PICARD_db].[dbo].[REPORT] | [account_rqst_id] | 
[PICARD_db].[dbo].[REPORT] | [originating_party_id], [archive_sts] | 
[PICARD_db].[dbo].[PROCESS_OBJECT_HISTORY] | [PROCESS_EVENT_NAME] | 
[PICARD_db].[dbo].[REPORT] | [originating_party_id] | 
[PICARD_db].[dbo].[REPORT] | [originating_party_id], [archive_sts] | 
[PICARD_db].[dbo].[PARTY] | [PRTY_STS] | 
[PICARD_db].[dbo].[REPORT] | [originating_party_id] | 
[PICARD_db].[dbo].[REPORT] | [originating_party_id], [archive_sts] | 
[PICARD_db].[dbo].[ACCOUNT_INFO] | [ACCOUNT_RQST_ID] | 
[PICARD_db].[dbo].[REPORT] | [tran_nbr], [reg_type_code] | 

**Duplicate Indexes**

There are times when an index is created with the same leading keys as another index. When indexes have the same key columns in the same order, they are duplicates. Additionally, indexes that share the same leading column order but do not share the same total columns are considered overlapping and could possibly be consolidated. Listed here are indexes that can be considered partial duplicates.

<table>
<thead>
<tr>
<th>Table</th>
<th>Index</th>
<th>Partial Duplicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo.PARTY_EXTENSION</td>
<td>PK__PARTY_EXTENSION__1DB2CC54</td>
<td>NonClusteredIndex-20130801-075027</td>
</tr>
<tr>
<td>dbo.WOTC_DATA</td>
<td>PK__WOTC_DATA__57478927237692A2</td>
<td>NonClusteredIndex-20150612-081343</td>
</tr>
<tr>
<td>dbo.SCRPT_DTA_OBJ_VAR</td>
<td>PK__SCRPT_DTA_OBJ_VAR__6F2B50E7</td>
<td>ix_Scrpt_Dta_Obj_Var_DOC_ID</td>
</tr>
<tr>
<td>dbo.WOTC_CREDIT</td>
<td>IX_WOTC_CREDIT</td>
<td>IX_WOTC_CREDIT_1</td>
</tr>
<tr>
<td>dbo.WOTC_CREDIT</td>
<td>IX_WOTC_CREDIT_2</td>
<td>IX_WOTC_CREDIT_3</td>
</tr>
<tr>
<td>dbo.PRTY_ASSN</td>
<td>PK__PRTY_ASSN__58FC18A6</td>
<td>SBJCT_PRTY_ID</td>
</tr>
<tr>
<td>dbo.wotc_data_old</td>
<td>PK__WOTC_DATA_new</td>
<td>IX_WOTC_DATA</td>
</tr>
<tr>
<td>dbo.wotc_data_old</td>
<td>IX_WOTC_DATA_2</td>
<td>IX_WOTC_DATA_3</td>
</tr>
<tr>
<td>dbo.CUSTOMER_REG</td>
<td>RecommendedIndex1</td>
<td>RecommendedIndex2</td>
</tr>
<tr>
<td>dbo.VARIABLE</td>
<td>PK__VARIABLE__5C63608D</td>
<td>IX_VARIABLE</td>
</tr>
<tr>
<td>dbo.VARIABLE</td>
<td>PK__VARIABLE__5C63608D</td>
<td>ix_Variable_VAR_ID</td>
</tr>
<tr>
<td>dbo.VARIABLE</td>
<td>IX_VARIABLE</td>
<td>ix_Variable_VAR_ID</td>
</tr>
<tr>
<td>dbo.DASHCOLS</td>
<td>PK_DASHCOLS</td>
<td>HOW_WE_GRAB_ALL_FOR_VIEW</td>
</tr>
</tbody>
</table>
Analysis

The following findings are category specific and include detailed descriptions:

Configuration

Finding: Compatibility level 90 for PICARD_db

The PICARD_db is running at compatibility level 90, meaning that the database is using the rules and features of SQL 2005. As this is not the current version of the SQL Server instance that is hosting the database, it adds additional overhead to process queries due to the compatibility translation. Additionally, it means your database can not take advantage of newer features available in SQL Server. It is a low risk effort to upgrade the compatibility level, which will provide a fundamental support and performance improvement.

❖ Action: Update compatibility level to 100 (2008 R2)

Finding: Filegrowth settings are set to percentage

The PICARD_db and tempdb databases have several files that are configured to grow by a percent. Use of percentage growth settings for SQL Server can lead to unpredictable growth of the database files. There can also be a performance impact, as SQL Server queries will wait when a file needs to be grown. It is recommended that explicit file growth be set relevant to the size of the database.

❖ Action: Set explicit filegrowth settings for data and log files
❖ Action: Ensure TempDB filegrowth settings are the same for all data files.

Finding: Successful log backup messages in SQL Server error log

The SQL Server error log is a key resource for identifying and troubleshooting issues in the instance. To make it easier to spot when problems are occurring, it should be kept free of non-critical messages as much as possible. By default, every successful backup operation adds an entry in the SQL Server error log and in the system event log. If you create very frequent log backups, these success messages accumulate quickly, resulting in huge error logs in which finding other messages is problematic.

By enabling trace flag 3226, you can suppress these log entries. This is useful if you are running frequent log backups and if none of your scripts depend on those entries.

❖ Action: Enable trace flag 3226 to suppress successful log backup messages from the SQL Server error log.
Performance

Finding: Data and Log files stored on the same volume

The PICARD_db data and transaction log files are housed on the same disk volume (D:). In a typical OLTP environment the IO profile of a data file is composed of random IOs, split relatively evenly between read and write operations. The IO of the transaction log file, on the other hand, is predominantly sequential write operations. Housing data and transaction files on the same physical disk generally leads to IO contention and a degradation in write performance to the transaction log file. Because of the critical role the transaction log plays in overall database performance, best practices from Microsoft recommend separating out this traffic between multiple volumes.

❖ **Action:** Create second volume to house log volumes

Finding: Inefficient storage configuration

Currently the D drive (PICARD_db data and transaction log files) is nonaligned with a starting offset of 32KB. Additionally, the Z drive (TempDB data and transaction log files) is formatted with a 4KB allocation unit size.

All database LUNs should have the allocation unit size (partition cluster size) set to 64Kb (65,536 bytes) for all partitions. Doing so will improve sequential write performance at a minimum and may reduce the split I/O (inefficient I/O) count. If partitions are backed up, destroyed, and then recreated on a Windows Server 2008 (or higher) system, the OS will automatically align the partitions to 1024KB. However, the cluster size for the partition will still need to be manually set to 64Kb (65,536 bytes). Note: this step requires new LUNs to be created in advance.

❖ **Action:** Move data files and tempdb to volumes that are properly aligned to 1024KB and configured with a 64KB allocation unit size.

Finding: Transaction log has a high number of VLFs

The PICARD_db log is comprised of 892 Virtual Log Files (VLFs). When the transaction log grows (either manually or via auto-growth), the new space allocated for the log is made up of virtual logs. When the log grows frequently, a large number of virtual logs can exist, which can cause a performance issue during startup, restores, and even inserts/updates/deletes. It is recommended to backup the log and then shrink it to reduce the number of VLFs. Then, grow the file to the appropriate size in 8000MB increments. The “appropriate size” for the transaction log will depend on the level of activity on the database and how frequently the log is being backed up.

❖ **Action:** Perform a transaction log backup, shrink the transaction log file to 0, and manually grow the file to the appropriate size. For log files larger than 8GB, grow the file in 8000MB increments.
Finding: No scheduled index or statistics maintenance

As data is inserted into tables and indexes, SQL Server creates additional pages to store this data in. These new pages are not always efficiently stored for access, creating fragmentation in the database. Over time, this fragmentation can have an impact on performance. Microsoft best practices are to run regular index maintenance operations to rebuild or reorganize fragmented indexes. A maintenance job to reorganize all indexes in the PICARD_db database exists, however it has not been executed since July of 2015.

There are 2 ways of defragmenting indexes in SQL Server, rebuilding and reorganizing. The index rebuild process is highly I/O intensive and generates a large amount of log activity. In addition, when an index is rebuilt offline, as is the case in the SG Heavy Industries environment, it will also lock the underlying table for the duration of the rebuild process. Performing this operation during peak periods can cause long-term blocking and deadlocks, and be highly disruptive to transactional processing. However, rebuilding an index is the most effective way to correct high-level fragmentation. Index reorganization is an online and relatively low-impact operation which is appropriate for low levels of fragmentation.

Because of the potential overhead, determining which indexes are rebuilt and when should be carefully considered. The level of fragmentation in an index should always dictate whether a rebuild is necessary. Microsoft recommends that indexes with a fragmentation level >5% and <= 30% be reorganized. Indexes more than 30% fragmented should be rebuilt. Bear in mind that these values are general recommendations. Depending on the size of the objects or the time available in your maintenance window, you may wish to raise those thresholds in order to be more selective in the rebuilds being performed. Note that smaller indexes (1000 pages or less) and indexes with very low levels of fragmentation (<=5%) should not be addressed, as they generally do not see a benefit from rebuilding.

In an environment such as SG Heavy Industries, we would recommend performing index maintenance on a weekly basis, preferably during a less active period when the impact on transactional processing will be minimized. For databases that are backed up weekly, with daily differential backups, the index maintenance should be performed prior to the full backup, so as to minimize the size of subsequent differentials.

However, it is the accuracy of statistics, rather than the level of index fragmentation, which has the biggest impact on query performance. SQL Server relies heavily on statistics regarding the data distribution in a table to form an optimal query plan. Outdated statistics lead to poor performance. To handle this, we would recommend creating a job to update statistics on a daily basis. Updating statistics is a low-impact,
online operation that will not block transactional activity. Overhead can be further reduced by updating statistics only where data has changed. For very volatile tables, it may be desirable to create additional jobs to update statistics on those tables more frequently. Note that index statistics are updated when an index is rebuilt (but not reorganized), so following an index rebuild with a statistics update is unnecessary. However, it is still necessary to ensure that column statistics are updated, as well.

- **Action:** Create and schedule a weekly job to reorganize indexes with >5% and <=30% fragmentation, and rebuild indexes with >30% fragmentation. Limit the procedure to indexes with more than 1000 pages (not based on row count).
- **Action:** Create and schedule a daily job to update modified index and column statistics with a 100% sample.

**Finding: High CXPacket Wait Type**

CXPACKET accounts for 77.4% of the CXPACKET indicates a heavy use of parallelism in queries running on the SQL Server. This could be the result of several different factors, including (but not limited to) improper parallelism configurations, inefficient queries, and a lack of indexing within the database. There is no single answer to addressing this issue, so recommended actions will cover several options.

- **Action:** Set the instance level MAXDOP setting to 8
- **Action:** Set the instance level Cost Threshold for Parallelism setting to 65
- **Action:** Perform query tuning analysis on missing indexes and top poorly performing queries to identify candidates for improvement.

**Finding: Missing index on REPORT.account_rqst_id**

Numerous high-read and high-duration queries captured during the assessment included a search predicate on the account_rqst_id. Creating an index on this column will help to eliminate full scans on the dashboard table or related indexes and facilitate seek operations instead, thereby dramatically reducing the IO incurred by each execution of these queries. It is notable that this is also the most recommended index based on SQL Server’s internal tracking mechanism.

- **Action:** Create a nonclustered index on REPORT.account_rqst_id

**Finding: Use of “SELECT *” with no predicate**

The three queries with the highest average and total read operations captured over an 8-hour period during the assessment are shown in the table below.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Executions</th>
<th>Avg Reads</th>
<th>Avg Writes</th>
<th>Avg Duration(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM ORDERS_TABLE</td>
<td>73</td>
<td>8,425,861</td>
<td>13,568</td>
<td>7,524</td>
</tr>
</tbody>
</table>
It appears that a cursor is being used to repeatedly read each of these tables, causing a large amount of read IO. Wherever possible, it is recommended to select only the columns needed to satisfy the applications needs, and to include a predicate to limit the number of rows returned, thereby minimizing IO operations.

- **Action:** Review code to ensure only relevant data is being retrieved to avoid unnecessary IO.

### Disaster Recovery

**Finding: Integrity Checks currently not being performed**

Database integrity checks (commonly referred to as CHECKDB), check the logical and physical structures of a database for possible signs of corruption. The earlier corruption is detected, the greater the chance of recovering from that corruption with minimal data loss. Therefore, it is critical to perform integrity checks on all databases, including system databases, on a regular basis. The amount of I/O generated by an integrity check can be extremely high (the entire database must be read into memory) and it is recommended that these checks be performed during off hours, at least once a week.

- **Action:** Create a job or set of jobs to check database integrity on a daily or weekly basis.

**Finding: Page Verify setting for PICARD_db is NONE**

SQL Server uses several internal mechanisms to validate the data stored on disk. The Page Verify setting defines how SQL Server will check for both logical and physical corruption within the data pages. When set to NONE, this means SQL Server is not performing any of these checks. Changing this is low risk and gives SQL Server the ability to detect and alert for corruption.

- **Action:** Change Page Verify for PICARD_db to CHECKSUM

**Finding: Master, Model, and MSDB system databases stored on C:\**

By storing the system databases on the OS volume, it is possible that SQL Server will halt if the OS drive fills up. A secondary risk is any I/O operations done by the OS will also contend with these system databases. It is a best practice to place the system databases on to another volume, either their own or the volumes for the user database files.

- **Action:** Move system databases to a different volume
Finding: Default backup path targets the C:\

Having the default backup path targeting the OS volume introduces the risk that a backup could accidentally be made to that volume. Database backups vary in size, but a large enough backup would fill up the OS volume and potentially cause SQL Server to halt. It is recommended that the default backup drive be set to the current backup volume for the instance.

 Action: Change default backup path to E:\Backups

Security

Finding: Logins and users with elevated privileges

At the time of the assessment every login on the ENTERPRISE instance was a member of the sysadmin server role and the corresponding user was a member of the db_owner role in the PICARD_db database. This means that every connection, including those from the web application, has the ability to perform any operation in the SQL Server instance.

 Action: Review existing logins and users to determine if the level of access currently assigned is appropriate.

Finding: BUILTIN\Administrators enabled

The BUILTIN\Administrators group is a login group that allows local Windows administrators on the host server access. The group has sysadmin level access in SQL Server, meaning that anyone within the local Windows Administrators group will have elevated SQL Server privileges. This is a concern because it makes the separation of access and control more difficult. It is recommended that this group be removed and sysadmin access be granted via another mechanism for better control.

 Action: Remove the BUILTIN\Administrators group and replace it with an Active Directory group dedicated for SQL Server administrators.

Finding: PICARD_db is not owned by SA

Database ownership grants complete control to a database and the objects within it. Unless specifically required, having a database owned by an individual login means that it cannot be secured completely against the login that owns it. Best practice is to set database ownership to SA within SQL Server.

 Action: Change database ownership to SA
Finding: SQL Server Agent Jobs not owned by SA

SQL Agent Jobs can only be edited and properly managed by either the logins that own them or logins with sysadmin level privileges. Because of this, it is generally recommended to set the ownership of Agent Jobs to SA. This allows for more consistent job management.

- **Action:** Set SQL Agent ownership to SA

Finding: xp_cmdshell enabled

Extended stored procedures are a set of code within SQL Server that allows for it to interact with components of the Windows environment outside of the database. The procedure xp_cmdshell allows DOS commands to be run from SQL Server. The issue is that these commands have the potential to execute as the SQL Server service account, which typically has elevated privileges in the OS. Because of this, it is considered a security risk. It is recommended that, unless there is a specific need for this functionality, it be disabled.

- **Action:** Disable xp_cmdshell

Additional Concerns

Finding: SQL Server 2008 R2 is not at the current service pack level

The SQL Server 2008 R2 installation is at Service Pack 2, Cumulative Update 13. Service Pack 3 is currently available. Service Packs provide all the combined updates as well as additional functionality improvements. In general, it is recommended to have the most recent Service Pack installed.

- **Action:** Upgrade installation to SQL Server 2008 R2 SP3
Conclusions

The performance metrics gathered during the assessment period indicate adequate processor and memory resources to support the current workload. While occasional spikes were recorded, disk read and write latency metrics, on average, were also within recommended parameters. It is believed that the recommendations put forward in this report will further improve performance and concurrency within the ENTERPRISE instance.

UpSearch provided the following rough order of magnitude time estimates to complete the aforementioned recommendations. The preceding time estimates by recommendation are provided as a courtesy and do not constitute a formal Statement of Work.

- Update SQL Server Configurations (backup locations, trace flags, parallelism settings)
  **Estimate:** 1 hour

- Update Database Configurations (compatibility levels, database file growth settings, database ownership, page verify settings)
  **Estimate:** 2 hours

- Update SQL Server 2008 R2 to Service Pack 3
  **Estimate:** 2 hours

- Reconfigure storage and relocate database files
  **Estimate:** 5 hours

- Install, configure, and validate database maintenance scripts (index maintenance, integrity checks)
  **Estimate:** 10 hours

- Security review and remediation for elevated logins and configurations
  **Estimate:** 10 hours

- Initial query tuning analysis and remediation
  **Estimate:** 20 hours
About the Authors

**Colleen Morrow** is Principal Consultant at *UpSearch.*

Colleen is a database strategist, community advocate, author, blogger and public speaker. She is passionate about helping technology leaders use Microsoft’s SQL Server to protect, optimize and unlock data’s value.

Colleen has been working with relational databases for almost 20 years. Since 2000, Colleen has specialized in SQL Server and Oracle database management solutions. She excels at performance tuning, troubleshooting mission critical SQL Server environments, and training technology professionals and business users of all levels.

Since 2011, Colleen has maintained a SQL Server focused blog at [http://colleenmorrow.com/](http://colleenmorrow.com/). She is Secretary of the Ohio North SQL Server User Group, as well as a volunteer for the Professional Association for SQL Server (PASS).

**Education & Certifications**

- Microsoft Certified Solutions Expert: Data Platform (MCSE: Data Platform)
- Azure Technical Assessments: Data Analytics & Data Platform Solutions
- Bachelor of Science in Computer and Information Systems, Cleveland State University

**Information Technology Skills**

- Microsoft Azure
- Microsoft SQL Server (through 2014)
- Microsoft SQL Server Reporting Services (SSRS)
- Oracle (9i through 11g)

**Publications**

- What Changed? Auditing Solutions in SQL Server, Tribal SQL: New voices in SQL Server, Chapter 7 ([http://www.amazon.com/Tribal-SQL-Tony-Davis-ebook/dp/B00H3JP4R0](http://www.amazon.com/Tribal-SQL-Tony-Davis-ebook/dp/B00H3JP4R0))

**Speaking & Community Engagements**

Colleen Morrow has presented at PASS Summit 2015, Enterprise Auditing with SQL Server Audit (Enterprise Database Administration & Deployment), PASS Summit 2014, SQL Audit: from Introduction to Automation (Enterprise Database Administration & Deployment), and at many SQL Saturday conferences and user groups throughout the United States.
David Maxwell is a SQL Server Consultant at UpSearch.

David Maxwell is a database strategist, community evangelist and public speaker. He is passionate about helping community and business leaders get the most out of SQL Server.

Since 2000, David has served to protect, unlock and optimize data’s value within such diverse environments as healthcare providers, insurance companies, manufacturers and financial institutions. David has the unique ability to zero in on a complex challenge quickly, and provide a long-term solution that fits both the business needs and budget.

Since 2012, David has maintained a SQL Server focused blog at https://dmmaxwell.wordpress.com/. He is a frequent presenter at local and regional events for the Professional Association for SQL Server (PASS) and works with the Columbus, OH based PASS chapter.

In his free time, David is an avid musician who plays several instruments, as well as a lover of puzzle-based games.

Education & Certifications

- Microsoft Certified Solutions Expert: Data Platform (MCSE: Data Platform)
- IEPTO1: Immersion Event on Performance Tuning and Optimization – Part 1
- IEPTO2: Immersion Event on Performance Tuning and Optimization – Part 2

Information Technology Skills

- Microsoft SQL Server (through 2014)
- Microsoft SQL Server Reporting Services (SSRS)
- Microsoft SQL Server Integration Services (SSIS)
- Systems architecture design, implementation, and administration
- TCP/IP Networking with DNS and DHC

Speaking & Community Engagements

Since 2012, David has presented sessions on SQL Server maintenance, performance and database corruption at numerous SQL Saturday events throughout the Midwest, in addition to being a presenter and director for his local Columbus, OH based PASS chapter. David has also presented for several online events such as the PASS DBA Fundamentals and Performance Virtual Chapters, and 24 Hours of PASS. He is also the winner of the PASS Summit 2015 Speaker Idol competition.
**Michael Fal** is a SQL Server Consultant at *UpSearch*.

Mike is a specialist in data management technologies. As a community advocate, public speaker, and blogger, Mike is a practicing thought leader for data and automation. He is passionate about DevOps and data, building platforms to optimize, protect, and use data efficiently.

Since 1999, Mike has been working in the database field, focusing primarily on SQL Server and specializes in automating data solutions to improve the reliability and efficiency of his environments. Mike has the unique ability to create robust, repeatable solutions to resolve a variety of technical challenges.

Mike has been working with databases for more than 15 years. He continues to blog at [http://mikefal.net](http://mikefal.net) since 2011, writing about a wide range of topics that center around data management. He is an active member, and former board member, of the Denver SQL Server User Group as well as a volunteer for the Professional Association for SQL Server (PASS).

**Education & Certifications**

- Microsoft Certified Solutions Expert: Data Platform (MCSE: Data Platform)
- Bachelor’s Degree, University of Colorado at Boulder

**Information Technology Skills**

- Microsoft Azure
- Amazon Web Services (AWS)
- Microsoft SQL Server (through 2014)
- Powershell
- Database Development & Design
- High Availability & Disaster Recovery
- DevOps & Data Management
- Performance Tuning
- NoSQL / Non-Relational Data Management
- DevOps & Data Management
- Performance Tuning
- NoSQL / Non-Relational Data Management

**Publications**

- *DevOps and the DBA*, Simple Talk by RedGate
- *A Month of SQLPS Blog Series*, Art Of The DBA
- Guest Posts – [Hey Scripting Guy TechNet Blog](http://mikefal.net)

**Speaking & Community Engagements**

Ben Miller is a Principal Consultant at UpSearch.

Ben is a database strategist, community advocate, author and public speaker. Ben’s passion to automate runs deep. He exemplifies how to work smarter and automates whenever possible in order to get more done with less effort.

Ben is a Microsoft Certified Master (SQL Server) and Microsoft Data Platform MVP. He holds numerous advanced industry certifications. Ben is also an active member of Professional Association for SQL Server (PASS) and leads the Utah County Chapter SQL Server User Group. He earned a Bachelor of Science in Information Technology from Western Governors University.

Ben is a lifelong bowler and data professional. He has been bowling since he was 12 and has bowled a sanctioned 300 game putting him in the Bowler’s Hall of Fame. He loves helping others get more out of life through education and volunteers regularly in his local area.

Awards

- Microsoft Data Platform MVP (2011 – Present)

Education & Certifications

- Microsoft Certified Master (MCM) : SQL Server 2008
- Microsoft Certified Solutions Master (MCSM) : Charter Data Platform
- Microsoft Certified Database Administrator (MCDBA)
- Microsoft Certified Applications Developer (MCAD) : .Net 2.0
- Bachelor of Science, Information Technology, Western Governors University

Information Technology Skills

- Microsoft Azure
- Amazon Web Services (AWS)
- Microsoft SQL Server Engine Internals
- Microsoft SQL Server Encryption
- Microsoft SQL Server Partitioning
- Microsoft SQL Server Availability Groups
- Microsoft SQL Server Clustering
- Data Warehouse & ETL Design
- Database Development & Design
- High Availability & Disaster Recovery
- DevOps & Data Management
- ETL Architecture
- Performance Tuning
- Powershell

Publications & Presentations

- SQL Server MVP Deep Dives, Vol. 2, Chapter 26 – SQL Server Filestream : To Blob or not to Blob
- PowerShell Deep Dives, Chapter 23 – SQL Server Provider in PowerShell
- SQL Server PowerShell Stairway, SQL Server Central

Speaking & Community Engagements

Ben has presented at many SQL Saturday conferences throughout the United States. He helped plan and organize Salt Lake City SQL Saturday events in 2012 and 2014.
About UpSearch


We offer the following services:

- SQL Server monitoring and companion DBA support, which bundles our own proprietary tools, *SQL Sentry’s Power Suite Plus* and a predetermined number of dedicated monthly support hours.

- *UpSearch’s* Block of Hours Model provides support against a pre-negotiated block of hours. The larger your annual commitment, the lower the bill rate. Our clients typically assemble twelve months’ worth of projects and support to lock-in the best bill rate.

Our team will complement your team across a broad spectrum of database initiatives, such as:

- Migrations to Azure
- Health Checks and Baseline Assessments
- High Availability & Disaster Recovery
- Database Design and Development
- DevOps and Data Management
- Performance Analysis and Tuning
- Business Intelligence
- Conversions to SQL Server
- Server Consolidations / Upgrades
- Virtualization

On-premises or in the cloud, *UpSearch* can help you protect, unlock and optimize your data’s value.