Quick Scan Report – High VLF Count

Virtual Log Files (VLFs) are part of the SQL Server log file. When space is allocated in the log due to growth, that new chunk of log is broken up into Virtual Log Files.

High VLF counts can slow down the following actions:

- Transaction Log Backup
- Crash Recovery, the process your database goes through on startup.
- Writing to the transaction log.

The interesting one here is crash recovery. If you have ever had a hard crash of the Windows OS and had to power off the server, you will often times see a database in “In Recovery”. What is happening at this point is that SQL Server is analyzing the transaction log files and verifying the integrity of the data in the database. I have seen this take several hours on databases with high VLF counts, compared to minutes on databases with a lower number of VLFs. Reducing the VLF count usually speeds up the crash recovery “In Recovery” time.

VLF Growth

VLFs are added in the following increments of log growth:

- up to 64mb = 4 vlfs
- 64mb up to 1gb = 8 vlfs
- More than 1gb = 16 vlfs

Visualizing VLFs

Here is a query to run on servers older than SQL Server 2012, such as SQL Server 2005, SQL Server 2008, and SQL Server 2008R2.

```
DECLARE @logInfoResults AS TABLE
(
 [FileId] TINYINT,
 [FileSize] BIGINT,
 [StartOffset] BIGINT,
 [FSeqNo] INTEGER,
 [Status] TINYINT,
 [Parity] TINYINT,
 [CreateLSN] NUMERIC(38,0)
);

INSERT INTO @logInfoResults
EXEC sp_executesql N'DBCC LOGINFO WITH NO_INFOMSGS';

SELECT FileSize / 1024 / 1024 as FileSizeInMB,
 [Status] ,
 REPLICATE('x', FileSize / 1024 / 1024 ) as [BarChart ______________________________
__________________________________________________________________]
 FROM @logInfoResults ;
```

Use the following Query on SQL Server 2012 or newer such as SQL Server 2012, SQL Server 2014, SQL Server 2016, SQL Server 2017, SQL Server 2019, SQL Server 2022 or newer to visualize the VLFs inside of your log file.

```
DECLARE @logInfoResults AS TABLE
(
[RecoveryUnitId] BIGINT, -- only on SQL Server 2012 and newer
[FileId] TINYINT,
[FileSize] BIGINT,
[StartOffset] BIGINT,
[FSeqNo] INTEGER,
[Status] TINYINT,
[Parity] TINYINT,
[CreateLSN] NUMERIC(38,0)
);

INSERT INTO @logInfoResults
```

```
    EXECUTE sp_executesql N'DBCC LOGINFO WITH NO_INFOMSGS';

SELECT
       DB_NAME(),
       CAST(FileSize / 1024.0 / 1024 AS DECIMAL(20,1)) AS FileSizeInMB,
       CASE
             WHEN FSeqNo = 0 THEN 'Available - Never Used'
             ELSE (CASE WHEN [Status] = 2 THEN 'In Use' ELSE 'Available' END)
       END AS TextStatus,
       [Status] ,
       REPLICATE( CASE WHEN [Status] = 2 THEN 'X' ELSE 'O' END , FileSize / MIN(File
Size) OVER())
             AS [BarChart __________________________________________________________
______________________________________]
FROM @logInfoResults ;
```

The way that you can reduce the VLF count is to back up the transaction log, shrink your transaction log file, then re-expand it to the size needed in the appropriate blocks to get the VLF’s to a reasonable number. I usually prefer to have less than 200 VLF’s.

Related Links

- Visualizing VLF sizing at SteveStedman.com
- Quick Scan Report