## Performance improvements in tempdb for SQL Server

Starting with SQL Server 2016 (13.x), **tempdb** performance is further optimized in the following ways:

* Temporary tables and table variables are cached. Caching allows operations that drop and create the temporary objects to execute very quickly and reduces page allocation contention.
* Allocation page latching protocol is improved to reduce the number of UP (update) latches that are used.
* Logging overhead for **tempdb** is reduced to reduce disk I/O bandwidth consumption on the **tempdb** log file.
* Setup adds multiple tempdb data files during a new instance installation. This task can be accomplished with the new UI input control on the **Database Engine Configuration** section and a command-line parameter /SQLTEMPDBFILECOUNT. By default, setup adds as many tempdb data files as the logical processor count or eight, whichever is lower.
* When there are multiple **tempdb** data files, all files autogrow at same time and by the same amount depending on growth settings. Trace flag 1117 is no longer required.
* All allocations in **tempdb** use uniform extents. [Trace flag 1118](https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-traceon-trace-flags-transact-sql?view=sql-server-2017) is no longer required.
* For the primary filegroup, the AUTOGROW\_ALL\_FILES property is turned on and the property cannot be modified.

## Capacity Planning for tempdb in SQL Server

Determining the appropriate size for tempdb in a SQL Server production environment depends on many factors. As described previously in this article, these factors include the existing workload and the SQL Server features that are used. We recommend that you analyze the existing workload by performing the following tasks in a SQL Server test environment:

* Set autogrow on for tempdb.
* Execute individual queries or workload trace files and monitor tempdb space use.
* Execute index maintenance operations, such as rebuilding indexes and monitor tempdb space.
* Use the space-use values from the previous steps to predict your total workload usage; adjust this value for projected concurrent activity, and then set the size of tempdb accordingly.