SQL Server Memory Usage (Percent) alert

The SQL Server Memory Usage (Percent) alert provides the total server memory (Total Server Memory per `sysperfinfo`) as a percent of total physical memory from WMI.

Reduce the percentage of memory used by SQL Server

If this value is regularly over 80%, SQL Server needs more memory or needs to use the memory it has more efficiently. Consider implementing one or more of the following solutions:

- If your site makes use of extended stored procedures that are infrequently called, then after calling them, issue a `DBCC DLLNAME (FREE)`. Once an extended stored procedure is called, it remains in memory until SQL Server is shut down even if it is never again called, which then wastes available memory. A `DBCC dllname (FREE)` releases that memory for use by both the procedure cache and Data Pages, which has a significant positive effect on both the Procedure Cache Hit Ratio and the Buffer Cache Hit Ratio. Execute `sp_helpextendedproc` to view the extended stored procedures currently loaded in memory by SQL Server.
- As each SQL Server lock requires 96 bytes of memory, the granting of lock space is done at the expense of Data Pages and Procedure Cache Pages. To maintain proper system performance and throughput, keep the number of locks to a minimum by:
  - Wherever possible, using the `(NOLOCK)` optimizer hint or `SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED` on select statements as this neither issues any shared locks on the data it reads nor honors any exclusive locks set by other transactions.
  - When updating all rows in a table with more than 50 rows, using the `TABLOCKX` table hint. This table hint prevents SQL Server from initially taking exclusive row locks, granting many of these locks, and then escalating them to an exclusive table lock.
  - When deleting all rows in any table, using the TRUNCATE TABLE statement instead of the DELETE statement as fewer locks and other system resources are consumed in the process.
  - Reducing the time that a lock is held by:
    - Performing as much work as possible before the transaction performs its first update, delete, or insert. For example, add any necessary SELECT statements.
    - Grouping all UPDATES, DELETES, and INSERTS as closely as possible within a transaction with as few SELECTS as possible separating them.
    - Committing the transaction as soon as possible after the final DML statement.
    - Avoiding any stops for user input once the transaction begins. Be sure to gather all user inputs before the transaction starts.
    - Allowing SQL Server to consume more of the available memory, making sure that OS Paging does not increase.
- Add more physical memory (RAM) to the computer.
- If the computer is running multiple instances of SQL Server, then consider placing each instance on a separate physical computer.
- If the computer is running other memory-intensive applications, such as IIS or Exchange, then consider moving each instance to a separate physical computer.
- Limit SQL Server computers to performing only SQL Server work. Stop any unnecessary programs, such as allowing the computer to act as either a primary or backup domain controller.

To enable alerting when this metric is outside its established baseline, click the **Baseline Thresholds Enabled (as percentage of baseline)** check box in the Alert Configuration window.