sys.dm_os_wait_stats (Transact-SQL)

Returns information about all the waits encountered by threads that executed. This aggregated view is used to diagnose performance issues with SQL Server and also with specific queries and batches. Also, shows the time for waits that have completed. This dynamic management view (DMV) does not show current waits but helps to understand wait stats.

Column name	Data type	Description
wait_type	**nvarchar(60)**	Name of the wait type.
waiting_tasks_count	**bigint**	Number of waits on this wait type. This counter is incremented at the start of each wait.
wait_time_ms	**bigint**	Total wait time for this wait type in milliseconds. This time is inclusive of signal_wait_time_ms.
max_wait_time_ms	**bigint**	Maximum wait time on this wait type.
signal_wait_time_ms	**bigint**	Difference between the time that the waiting thread was signaled and when it started running.

Types of Waits

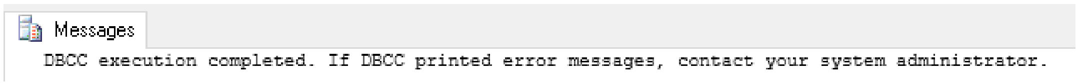
Resource waits - occur when a worker requests access to a resource that is not available because the resource is being used by some other worker or is not yet available. Examples: locks, latches, network and disk I/O waits. Lock and latch waits are waits on synchronization objects.

Queue waits - occur when a worker is idle, waiting for work to be assigned. Are most typically seen with system background tasks (i.e. the deadlock monitor, deleted record cleanup tasks). These tasks will wait for work requests to be placed into a work queue. Queue waits may also periodically become active even if no new packets have been put on the queue.

External waits - occur when a SQL Server worker is waiting for an external event (i.e. an extended stored procedure call, a linked server query) to finish. In the diagnose blocking issues, that external waits do not always imply that the worker is idle, because the worker may actively be running some external code.

The contents of this dynamic management view (DMV) can be reset:

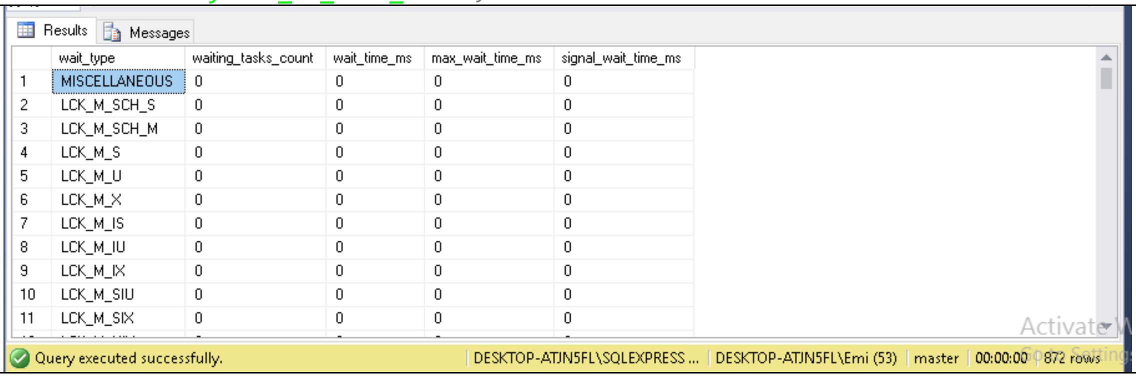
```sql
-- reset the values in the sys.dm_os_wait_stats DMV - resets all counters to 0.
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR);
GO
```



```
Messages
  DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

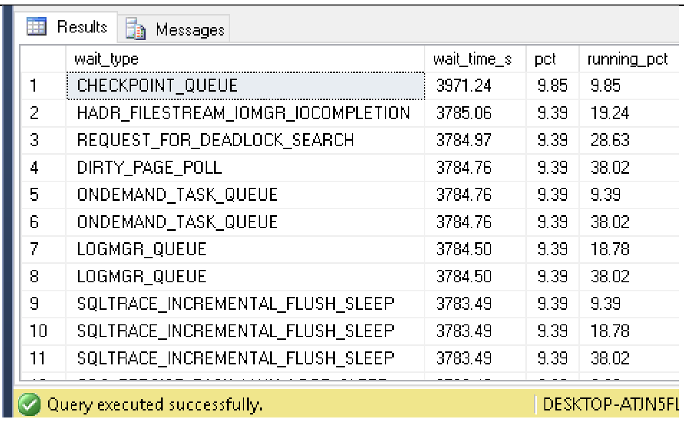
```sql
-- the data from this DMV (Dynamic Management Views) look like
SELECT * FROM sys.dm_os_wait_stats;
```



```
Results | Messages
    wait_type       waiting_tasks_count  wait_time_ms  max_wait_time_ms  signal_wait_time_ms
1   MISCELLANEOUS   0                    0             0                 0
2   LCK_M_SCH_S     0                    0             0                 0
3   LCK_M_SCH_M     0                    0             0                 0
4   LCK_M_S         0                    0             0                 0
5   LCK_M_U         0                    0             0                 0
6   LCK_M_X         0                    0             0                 0
7   LCK_M_IS        0                    0             0                 0
8   LCK_M_IU        0                    0             0                 0
9   LCK_M_IX        0                    0             0                 0
10  LCK_M_SIU       0                    0             0                 0
11  LCK_M_SIX       0                    0             0                 0
Query executed successfully. | DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (53) | master | 00:00:00 | 872 rows
```

By creating a common table expression to build a Waits table one can filter out irrelevant wait types and then look at a listing of just those waits that comprise the top N% (in this case 95%) of the waits on the SQL Server instance:

```
-- get a snapshot of waits in percentage form at the
current point in time
WITH Waits AS
 (
 SELECT  wait_type, wait_time_ms / 1000. AS wait_time_s,
   100. * wait_time_ms / SUM(wait_time_ms) OVER() AS pct,
   ROW_NUMBER() OVER(ORDER BY wait_time_ms DESC) AS rn
 FROM sys.dm_os_wait_stats
 WHERE wait_type NOT IN
     ('CLR_SEMAPHORE', 'LAZYWRITER_SLEEP',
'RESOURCE_QUEUE',
   'SLEEP_TASK', 'SLEEP_SYSTEMTASK',
'SQLTRACE_BUFFER_FLUSH', 'WAITFOR',   'CLR_AUTO_EVENT',
'CLR_MANUAL_EVENT')
   ) -- filter out additional irrelevant waits
  SELECT W1.wait_type,
 CAST(W1.wait_time_s AS DECIMAL(12, 2)) AS wait_time_s,
 CAST(W1.pct AS DECIMAL(12, 2)) AS pct,
 CAST(SUM(W2.pct) AS DECIMAL(12, 2)) AS running_pct
FROM Waits AS W1 INNER JOIN Waits AS W2 ON W2.rn <= W1.rn
GROUP BY W1.rn, W1.wait_type, W1.wait_time_s, W1.pct
HAVING SUM(W2.pct) - W1.pct < 95; -- percentage
threshold;
```

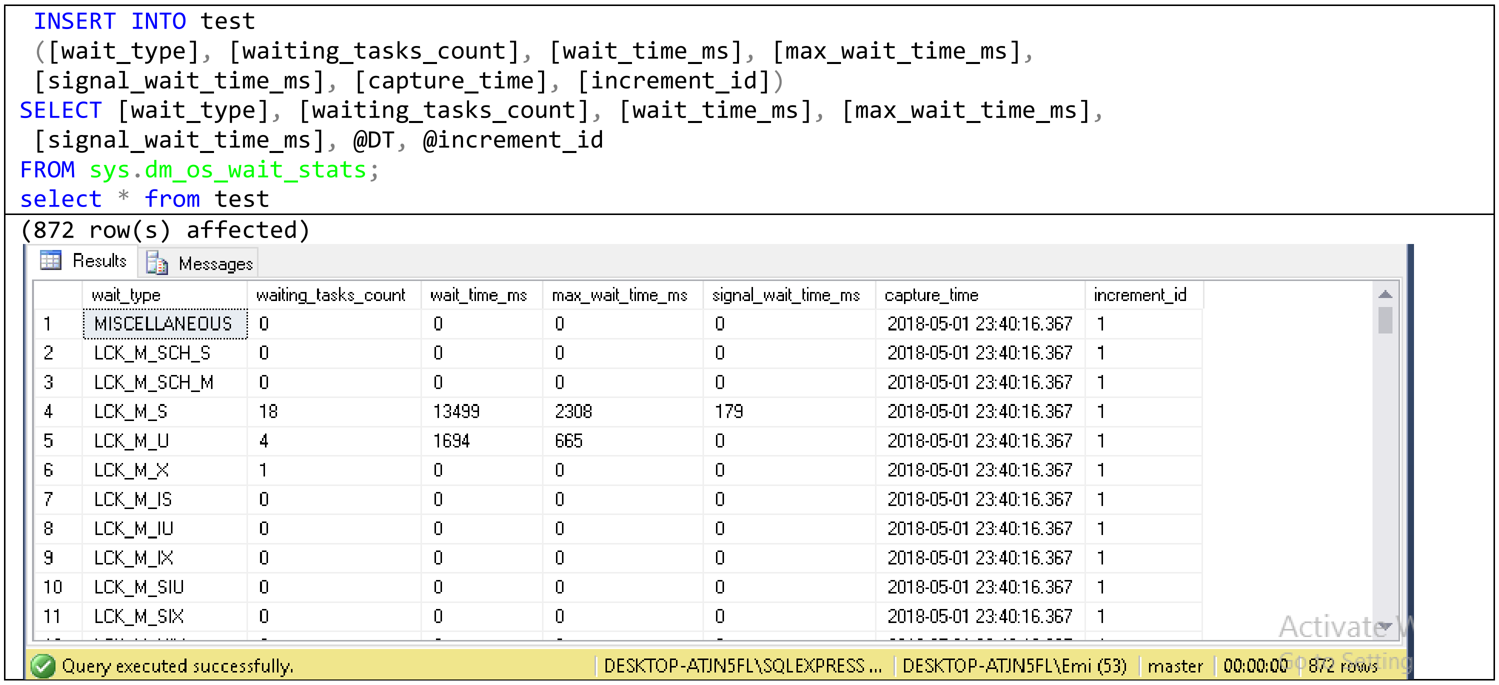


	wait_type	wait_time_s	pct	running_pct
1	CHECKPOINT_QUEUE	3971.24	9.85	9.85
2	HADR_FILESTREAM_IOMGR_IOCOMPLETION	3785.06	9.39	19.24
3	REQUEST_FOR_DEADLOCK_SEARCH	3784.97	9.39	28.63
4	DIRTY_PAGE_POLL	3784.76	9.39	38.02
5	ONDEMAND_TASK_QUEUE	3784.76	9.39	9.39
6	ONDEMAND_TASK_QUEUE	3784.76	9.39	38.02
7	LOGMGR_QUEUE	3784.50	9.39	18.78
8	LOGMGR_QUEUE	3784.50	9.39	38.02
9	SQLTRACE_INCREMENTAL_FLUSH_SLEEP	3783.49	9.39	9.39
10	SQLTRACE_INCREMENTAL_FLUSH_SLEEP	3783.49	9.39	18.78
11	SQLTRACE_INCREMENTAL_FLUSH_SLEEP	3783.49	9.39	38.02

Query executed successfully. DESKTOP-ATJN5FL

Since the last SQL Server services restart, one primarily dealing with ASYNC_NETWORK_IO and OLEDB waits on SQL Server instance. There are issues with a specific application on the server that causes these waits due to performing row-by-row processing of SQL batch results sets returned to the application. The SOS_SCHEDULER_YIELD wait occurs whenever a task offers up its place in queue to allow another process to run in its stead. It is indicative of CPU issues that may need to be addressed. Specific waits point in the direction of where to focus the tuning resources because the cause for certain waits are attributed directly to CPU, memory, or I/O.

```
-- query the information just for the period between the last two stats collections
       --USE [<database_name,,Foo>];
USE master
--Create table to persist wait stats information:
DROP TABLE dbo.test
CREATE TABLE dbo.test(
 [wait_type] [nvarchar](60) NOT NULL,
 [waiting_tasks_count] [bigint] NOT NULL,
 [wait_time_ms] [bigint] NOT NULL,
 [max_wait_time_ms] [bigint] NOT NULL,
 [signal_wait_time_ms] [bigint] NOT NULL,
 [capture_time] [datetime] NOT NULL,
 [increment_id] [int] NOT NULL
);
       ALTER TABLE test
 ADD  DEFAULT (GETDATE()) FOR [capture_time];
--Insert wait stats info in a datestamped format for later querying:
DECLARE @DT DATETIME ;
SET @DT = GETDATE() ;
DECLARE @increment_id INT;
SELECT @increment_id = MAX(increment_id) +1 FROM test
SELECT @increment_id = ISNULL(@increment_id, 1)
```

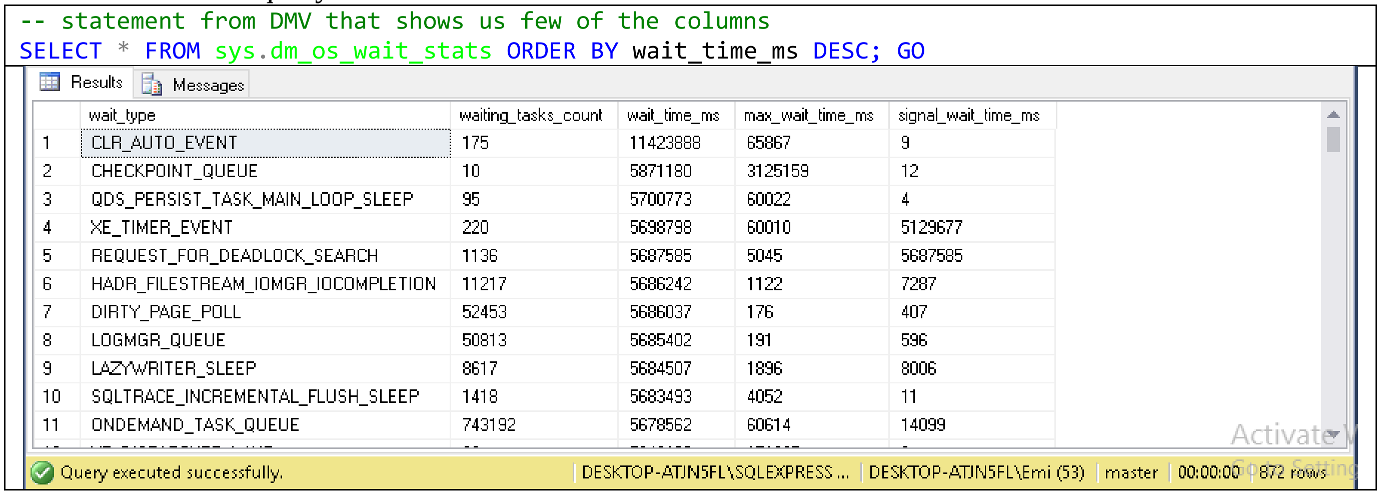


```sql
 INSERT INTO test
 ([wait_type], [waiting_tasks_count], [wait_time_ms], [max_wait_time_ms],
 [signal_wait_time_ms], [capture_time], [increment_id])
SELECT [wait_type], [waiting_tasks_count], [wait_time_ms], [max_wait_time_ms],
 [signal_wait_time_ms], @DT, @increment_id
FROM sys.dm_os_wait_stats;
select * from test
(872 row(s) affected)
```

	wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms	capture_time	increment_id
1	MISCELLANEOUS	0	0	0	0	2018-05-01 23:40:16.367	1
2	LCK_M_SCH_S	0	0	0	0	2018-05-01 23:40:16.367	1
3	LCK_M_SCH_M	0	0	0	0	2018-05-01 23:40:16.367	1
4	LCK_M_S	18	13499	2308	179	2018-05-01 23:40:16.367	1
5	LCK_M_U	4	1694	665	0	2018-05-01 23:40:16.367	1
6	LCK_M_X	1	0	0	0	2018-05-01 23:40:16.367	1
7	LCK_M_IS	0	0	0	0	2018-05-01 23:40:16.367	1
8	LCK_M_IU	0	0	0	0	2018-05-01 23:40:16.367	1
9	LCK_M_IX	0	0	0	0	2018-05-01 23:40:16.367	1
10	LCK_M_SIU	0	0	0	0	2018-05-01 23:40:16.367	1
11	LCK_M_SIX	0	0	0	0	2018-05-01 23:40:16.367	1

Query executed successfully. | DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (53) | master | 00:00:00 | 872 rows

Specific types of wait times during query execution can indicate bottlenecks or stall points within the query. Similarly, high wait times, or wait counts server wide can indicate bottlenecks or hot spots in interaction query interactions within the server instance.



```sql
-- statement from DMV that shows us few of the columns
SELECT * FROM sys.dm_os_wait_stats ORDER BY wait_time_ms DESC; GO
```

	wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
1	CLR_AUTO_EVENT	175	11423888	65867	9
2	CHECKPOINT_QUEUE	10	5871180	3125159	12
3	QDS_PERSIST_TASK_MAIN_LOOP_SLEEP	95	5700773	60022	4
4	XE_TIMER_EVENT	220	5698798	60010	5129677
5	REQUEST_FOR_DEADLOCK_SEARCH	1136	5687585	5045	5687585
6	HADR_FILESTREAM_IOMGR_IOCOMPLETION	11217	5686242	1122	7287
7	DIRTY_PAGE_POLL	52453	5686037	176	407
8	LOGMGR_QUEUE	50813	5685402	191	596
9	LAZYWRITER_SLEEP	8617	5684507	1896	8006
10	SQLTRACE_INCREMENTAL_FLUSH_SLEEP	1418	5683493	4052	11
11	ONDEMAND_TASK_QUEUE	743192	5678562	60614	14099

Query executed successfully. | DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (53) | master | 00:00:00 | 872 rows

References:

https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-wait-stats-transact-sql?view=sql-server-2017

https://www.mssqltips.com/sqlservertip/1949/sql-server-sysdmoswaitstats-dmv-queries/

https://blog.sqlauthority.com/2011/02/03/sql-server-dmv-sys-dm_os_wait_stats-explanation-wait-type-day-3-of-28/

http://www.sqlservercentral.com/blogs/sqldbauk/2010/02/18/sys.dm_5F00_os_5F00_wait_5F00_stats/