

# Troubleshooting Blocking

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST

---

## Contents

- [Issue](#)
- [Investigation / Analysis](#)
  - [Analyze findings in ASC](#)
  - [Check for waits of type LCK\\_\\*](#)
  - [Investigate the blocked process report](#)
- [Mitigation - Recommendations to prevent blocking](#)
- [More Information](#)
- [Internal Doc Reference](#)

## Troubleshooting Blocking in the telemetry

### Issue

This article provides you with steps for analyzing blocking in the Azure telemetry.

See the related article [Blocking](#) for steps that you can provide to or go through together with the customer.

### Investigation / Analysis

#### Analyze findings in ASC

Get the server/database name and the exact period of the issue from the customer. Then run the ASC Troubleshooter for it. Check the Insights on the "Performance" tab and go to the "Performance - Blocking and Deadlocks" section for more details.

It will give you an initial idea about the impact of the issue, and if there is any combined scenario of blocking, deadlocks, and long-running transactions.

#### Check for waits of type LCK\_\*

Run the following Kusto query for the customer resource. To reduce the noise, it limits the result to waits that are longer than 100ms; you may have to adapt this to a different value:

```

let startTime = datetime(2022-09-27 14:00:00Z);
let endTime = datetime(2022-09-27 16:00:00Z);
let srv = "servername";
let db = "databasename";
MonDmCloudDatabaseWaitStats
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
| where LogicalServerName =~ srv
| where database_name =~ db
| where delta_wait_time_ms > 100 or delta_max_wait_time_ms > 100
//| where wait_type startswith "LCK"
| project TIMESTAMP, start_utc_date, end_utc_date, NodeName, AppName, LogicalServerName, ResourcePoolName, dat

```

Example output:

TIMESTAMP	...	AppName	Lo...	R...	d...	database_name	wait_type	wait_time_ms	delta_wait_time_ms	max_wait_time_ms	delta_max_wait_time_ms	signal_wait_time_ms	delta_signal_wait_time_ms
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	PAGEIOLATCH_SH	3197	3166	39	10	35	35
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	MEMORY_ALLOCATIO...	117	104	0	0	0	0
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	ASYNC_NETWORK_IO	1977	1871	204	184	14	14
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	WAIT_ON_SYNC_STATI...	5126	5126	2322	2322	0	0
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	LCK_M_IU	7145	7145	7145	7145	0	0
2022-09-27 14:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	LCK_M_IU	180422	173277	173276	166131	0	0
2022-09-27 15:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	LCK_M_IU	4184337	4003915	4003915	3830639	0	0
2022-09-27 15:...	..	ac4fb573f27d	we...	h...	9	AdventureWorks	PREEMPTIVE_XHTTP	144	144	47	47	0	0

This output shows you the symptoms of blocking caused by a long-running transaction (see steps in [More Information](#) below). Both `delta_wait_time_ms` and `delta_max_wait_time_ms` are increasing for lock type `LCK_M_IU`, which was the Intent-Update lock requested by an `UPDLOCK` query hint. Once the blocking transaction has ended, the blocking is resolved.

Proceed further with the investigation when you see `LCK_*` waits on `MonDmCloudDatabaseWaitStats` with "delta\_waits" showing relevant values, especially reaching up and above typical timeout thresholds like 15000 or 30000 (15s or 30s).

**For a graphical rendering**, you can run the following variation of the Kusto query from above. Also note the filtering option for specific waittypes:

```

let startTime = datetime(2022-09-27 14:00:00Z);
let endTime = datetime(2022-09-27 16:00:00Z);
let srv = "servername";
let db = "databasename";
MonDmCloudDatabaseWaitStats
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
| where LogicalServerName =~ srv
| where database_name =~ db
| where delta_wait_time_ms > 100 or delta_max_wait_time_ms > 100
//| where wait_type startswith "LCK"
| summarize sum(delta_wait_time_ms) by bin(end_utc_date, 5min), wait_type
| sort by end_utc_date asc nulls last
| project end_utc_date, wait_type, sum_delta_wait_time_ms
| render timechart

```

## Investigate the blocked process report

The blocked process report is captured in the `MonBlockedProcessReportFiltered` Kusto table. Use the following query to easily decode the relevant details from the report's XML content. Some of the important content is PII-

filtered though, like the SQL text in the inputbuffer. If the SQL text is needed, you can use the query\_hash values from the output below and get further information from Query Store.

```

let startTime = datetime(2022-09-27 14:00:00Z);
let endTime = datetime(2022-09-27 16:00:00Z);
let srv = "servername";
let db = "databasename";
let blockingchain=MonBlockedProcessReportFiltered
| where TIMESTAMP > startTime
| where TIMESTAMP < endTime
| where LogicalServerName =~ srv
| where logical_database_name =~ db
//| where AppName =~ "a4b395401f2d" //and NodeName =~ "DB.87"
| extend monitorLoop = extract("monitorLoop=\"([0-9]+)\"", 1, blocked_process_filtered, typeof(int))
| parse blocked_process_filtered with anystr3:string "<blocked-process>" blockee "</blocked-process>" discard1
| parse blocked_process_filtered with anystr4:string "<blocking-process>" blocker "</blocking-process>" discard
| extend blockee_session_id = extract("spid=\"([0-9]+)\"", 1, blockee, typeof(int))
| extend blockee_status = extract("status=\"(.*)\"", 1, blockee, typeof(string))
| extend blockee_waittime = extract("waittime=\"([0-9]+)\"", 1, blockee, typeof(int))
| extend blockee_trancount = extract("trancount=\"([0-9]+)\"", 1, blockee, typeof(int))
| extend blockee_lasttranstarted = extract("lasttranstarted=\"(.*)\"", 1, blockee, typeof(datetime))
| extend blockee_queryhash = extract("queryhash=\"(.*)\"", 1, blockee, typeof(string))
| extend blockee_isolationlevel = extract("isolationlevel=\"(.*)\"", 1, blockee, typeof(string))
| extend blockee_lastattention = extract("lastattention=\"(.*)\"", 1, blockee, typeof(datetime))
| extend blockee_lastattention = extract("lastattention=\"(.*)\"", 1, blockee, typeof(datetime))
| extend blockee_lastbatchstarted = extract("lastbatchstarted=\"(.*)\"", 1, blockee, typeof(datetime))
| extend blockee_lastbatchcompleted = extract("lastbatchcompleted=\"(.*)\"", 1, blockee, typeof(datetime))
| extend blockee_waitresource = extract("waitresource=\"(.*)\"", 1, blockee, typeof(string))
| extend blockee_lockMode = extract("lockMode=\"(.*)\"", 1, blockee, typeof(string))
| extend blockee_clientapp = extract("clientapp=\"(.*)\"", 1, blockee, typeof(string))
| extend blocker_session_id = extract("spid=\"([0-9]+)\"", 1, blocker, typeof(int))
| extend blocker_status = extract("status=\"(.*)\"", 1, blocker, typeof(string))
| extend blocker_waittime = extract("waittime=\"([0-9]+)\"", 1, blocker, typeof(int))
| extend blocker_trancount = extract("trancount=\"([0-9]+)\"", 1, blocker, typeof(int))
| extend blocker_queryhash = extract("queryhash=\"(.*)\"", 1, blocker, typeof(string))
| extend blocker_isolationlevel = extract("isolationlevel=\"(.*)\"", 1, blocker, typeof(string))
| extend blocker_lastattention = extract("lastattention=\"(.*)\"", 1, blocker, typeof(datetime))
| extend blocker_lastattention = extract("lastattention=\"(.*)\"", 1, blocker, typeof(datetime))
| extend blocker_lastbatchstarted = extract("lastbatchstarted=\"(.*)\"", 1, blocker, typeof(datetime))
| extend blocker_lastbatchcompleted = extract("lastbatchcompleted=\"(.*)\"", 1, blocker, typeof(datetime))
| extend blocker_clientapp = extract("clientapp=\"(.*)\"", 1, blocker, typeof(string))
| order by monitorLoop asc nulls last;
let leadblockers=
blockingchain
| join kind= rightanti (
    blockingchain
) on $left.monitorLoop == $right.monitorLoop and $left.blockee_session_id==$right.blocker_session_id
| extend lead_blocker_session_id = blocker_session_id
| distinct monitorLoop, lead_blocker_session_id;
leadblockers
| join kind= inner (
    blockingchain
) on $left.monitorLoop == $right.monitorLoop and $left.lead_blocker_session_id==$right.blocker_session_id
| order by TIMESTAMP asc nulls last
| project monitorLoop, originalEventTimestamp, AppName, LogicalServerName, ResourcePoolName, database_id, data
blocker_session_id, blocker_isolationlevel, blocker_queryhash=toupper(blocker_queryhash), blocker_status, b1
blockee_session_id, blockee_waitresource, blockee_lockMode, blockee_isolationlevel, blockee_queryhash=toupper
//, blocked_process_filtered

```

## Example output:

monitorLoop	originalEventTimestamp	AppName	LogicalServerName	ResourcePoolName	database_id	database_name	lead_blocker_session_id	duration	resource_owner_type	lock_mode	transaction_id
181858	2022-09-27 14:28:45.6841902	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	26565000	LOCK	IU	11390336
181862	2022-09-27 14:29:05.7050495	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	46643000	LOCK	IU	11390336
181866	2022-09-27 14:29:25.7370158	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	66674000	LOCK	IU	11390336
181870	2022-09-27 14:29:45.7521653	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	86690000	LOCK	IU	11390336
181874	2022-09-27 14:30:05.7872684	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	106721000	LOCK	IU	11390336
181878	2022-09-27 14:30:25.8309918	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	126768000	LOCK	IU	11390336
181882	2022-09-27 14:30:45.8653203	ac4fb573f27d	weholgeri	hlipool	9	AdventureWorks	71	146799000	LOCK	IU	11390336



blocker_session_id	blocker_isolationlevel	blocker_queryhash	blocker_status	blocker_trancount	blocker_lastbatchstar...	blocker_lastbatchco...	blocker_lastattention	blocker_clientapp
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query
71	read committed (2)		sleeping	1	2022-09-27 14:26:23...	2022-09-27 14:26:23...	1900-01-01 00:00:00...	Microsoft SQL Server Management Studio - Query

blockee_session_id	blockee_waitresource	blockee_lockMode	blockee_isolationlevel	blockee_queryhash	blockee_waittime	blockee_status	blockee_trancount	blockee_lasttranstarted	blockee_lastbatchstar...	blockee_lastbatchco...	blockee_clientapp
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	26621	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	46645	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	66676	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	86692	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	106723	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	146801	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query
76	PAGE: 91:28632	IU	read committed (2)	0X43CD4686DA9886CB	126770	suspended	0	2022-09-27 14:28:19...	2022-09-27 14:28:19...	2022-09-27 14:28:19...	Microsoft SQL Server Management Studio - Query

## Mitigation - Recommendations to prevent blocking

Here are some recommendations to help minimizing the impact of blocking:

- Identify the query behind the head blocker and tune this query so that it runs faster and allocates less resources. Use the usual query tuning tools to avoid large scans, slow I/O, tempdb usage etc.
- Change the MAXDOP setting from 0 to a value between 1 and 8 (to avoid NUMA issues)
- Prevent lock escalation - see [Lock Escalation](#) for further details.
- Keep transactions as short as possible and avoid user interactions that prolong transaction duration.
- Look for opportunities to create new indexes or to add columns to an existing index to remove index or table scans and to maximize the efficiency of index seeks.
- Explore creating partitions on the tables so that queries can run against different partitions. See [Create partitioned tables and indexes](#) 
- Explore Columnstore indexes for the table. See [Columnstore indexes - Design guidance](#) 

Also see the "Mitigation" section in article [Blocking](#) for additional steps.

## More Information

If you want to test and reproduce a blocking issue, you need two connections to your database, e.g. two query windows in SSMS. Then perform actions similar to the following example from AdventureWorks (it works the same way with any table):

```
-- connect to AdventureWorks database

-- connection 1: create head blocker by keeping an active transaction
begin tran
update person.person set FirstName = 'Arnold' where FirstName = 'Kevin'
-- rollback

-- connection 2: blocked statement, requesting conflicting locks on the same rows
select LastName from person.person with (updlock) where FirstName = 'Kevin'
```

## Internal Doc Reference

- Additional variations of Kusto queries are available at [Blocking\\_\(Managed Instance\)](#).

**How good have you found this content?**

