Temp DB - Troubleshoot Tempdb from the telemetry

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

Contents

Issue

This article contains information about retrieving tempdb-related details from the Kusto telemetry.

For steps to troubleshoot tempdb issues from the customer side, see Temp DB - Resolve tempdb related errors and exceptions.

Investigation / Analysis

The first step always is to check the "Performance" tab on ASC to see the Insights and to get a general idea about the overall consumption.

Kusto - MonWiQdsExecStats

This Kusto query returns individual queries and their tempdb usage. It allows you to see each query based on its query_hash and the execution plan it was using.

The query includes two 2 "| project" lines: the first returns general troubleshooting information with detailed performance statistics, the second narrows down on tempdb. It also includes the "statement_type" column to see if this is a select, insert, update, delete, bulk operation.

```
let startTime = datetime(2022-10-12 06:30:00Z);
let endTime = datetime(2022-10-12 08:00:00Z);
let srv = "servername";
let db = "databasename";
MonWiQdsExecStats
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
| where LogicalServerName =~ srv
| where database_name =~ db
| where tempdb_space_used > 0 // filter on tempdb usage
//| where statement_type in ("x_estypInsertBulk", "x_estypInsert")
//| where query_hash == "0xEA234692500D5BDA"
| extend Average_cpu_time = cpu_time / execution_count,
         Average_logical_reads = logical_reads / execution_count,
         Average_logical_writes = logical_writes / execution_count,
         Average_physical_reads = physical_reads / execution_count,
         Average_elapsed_time = elapsed_time / execution_count,
         Average_log_bytes_used = log_bytes_used / execution_count,
         Average_rowcount = rowcount / execution_count
| project TIMESTAMP, AppName, NodeName, LogicalServerName, database_name, query_id, plan_id, query_hash, query
| project TIMESTAMP, LogicalServerName, database_name, query_hash, query_plan_hash, statement_type, exec_type,
// exec_type != 0 indicates failed queries
// exec_type: 0=regular, 3=timeout/aborted, 4=exception
```

Sample output:

TIMESTAMP	LogicalSe...	database_name	query_hash	query_plan_hash	statement_type	exec_type	execution_count	tempdb_space_used	max_tempdb_space_used	min_tempdb_space_used	rowcount	max_rowcount	Aver
2022-10-12 07:02:24....	weholgerl	AdventureWorks	0x6F10F71062CB980D	0x293E74CBFE1...	x_estypSelect	0	2	8	8	0	182	91	
2022-10-12 07:02:24....	weholgerl	AdventureWorks	0xFF021751A0FD3C33	0x8271C936C2C...	x_estypSelect	0	1	40	40	40	71	71	
2022-10-12 07:02:24....	weholgerl	AdventureWorks	0x05AC29686DFB144C	0xD2DA8CA422...	x_estypSelect	0	5	8	8	0	45	16	
2022-10-12 07:02:24....	weholgerl	AdventureWorks	0x3231BC4D6A5380BC	0xEE38276150A...	x_estypSelect	0	12	16	16	0	3	1	
2022-10-12 07:02:24....	weholgerl	AdventureWorks	0xF2E352DF17E31CAA	0xFBF5C464A9A...	x_estypSelect	0	6	16	16	0	2	1	
2022-10-12 07:17:24....	weholgerl	AdventureWorks	0xEA234692500D5BDA	0x00024F9031F...	x_estypSelect	3	2	232	120	112	0	0	

Kusto - MonDmIoVirtualFileStats

This Kusto query will show you the change in space consumption in your user database and in tempdb over time. The granularity is 1 hour, so if there is a massive growth and shrink between the capture times, you won't see it on the telemetry. It will however give you a good overview about trends and patterns.

```
let startTime = datetime(2022-10-12 06:30:00Z);
let endTime = datetime(2022-10-12 08:00:00Z);
let srv = "servername";
let db = "databasename";
MonDmIoVirtualFileStats
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
| where LogicalServerName =~ srv
| where db_name in ("tempdb", db)
| extend size_on_disk_mb=(size_on_disk_bytes*1.0/1024/1024)
| extend max_space_used_percent = toint((1.0 * spaceused_mb / max_size_mb) * 100)
| project TIMESTAMP, NodeName, AppName, LogicalServerName, db_name, database_id, file_id, type_desc, is_primar
| order by TIMESTAMP asc nulls last, db_name, file_id asc nulls last
```

Sample output:

TIMESTAMP	Node...	AppName	LogicalServer...	db_name	database_id	file_id	type_desc	is_primary_...	is_remote	storage_type	growth	spaceused_mb	size_on_disk_mb	max_size_mb	max_space_used_percent
2022-10-12 06:55:13....	_DB_12	e4aeb2328e63	weholgerl	tempdb	2	1	ROWS	0	0	8	8192	9	16	14225	0
2022-10-12 06:55:13....	_DB_12	e4aeb2328e63	weholgerl	tempdb	2	2	LOG	0	0	8	8192	3	16	6758	0
2022-10-12 06:55:13....	_DB_12	e4aeb2328e63	weholgerl	AdventureWorks	9	1	ROWS	1	1	1	2048	240	368	2048	11
2022-10-12 06:55:13....	_DB_12	e4aeb2328e63	weholgerl	AdventureWorks	9	2	LOG	1	1	1	2048	2	264	5120	0
2022-10-12 07:55:13....	_DB_12	e4aeb2328e63	weholgerl	tempdb	2	1	ROWS	0	0	8	8192	10	16	14225	0
2022-10-12 07:55:13....	_DB_12	e4aeb2328e63	weholgerl	tempdb	2	2	LOG	0	0	8	8192	4	16	6758	0
2022-10-12 07:55:13....	_DB_12	e4aeb2328e63	weholgerl	AdventureWorks	9	1	ROWS	1	1	1	2048	241	368	2048	11
2022-10-12 07:55:13....	_DB_12	e4aeb2328e63	weholgerl	AdventureWorks	9	2	LOG	1	1	1	2048	1	264	5120	0

Kusto - AlrSQLErrorsReported and MonSQLSystemHealth

These Kusto tables are the equivalent of the SQL Server errorlog. Check for errors that are related to running out of space. There usually is nothing returned here - but if it is, check the output if the error is related to the user database or tempdb.

```
// explicitly-reported errors
let startTime = datetime(2022-10-12 06:30:00Z);
let endTime = datetime(2022-10-12 08:00:00Z);
let srv = "servername";
let db = "databasename";
let app = "e4aeb2328e63";
AlrSQLErrorsReported
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
| where LogicalServerName =~ srv
//| where database_name =~ db
//| where AppName =~ app
//| where error_number in (1101, 1105, 9001, 9002)
| project originalEventTimestamp, database_name, AppName, NodeName, error_number, severity, state, category, d
| limit 1000

// ERRORLOG details
let startTime = datetime(2022-10-12 06:30:00Z);
let endTime = datetime(2022-10-12 08:00:00Z);
let srv = "servername";
let app = "e4aeb2328e63";
MonSQLSystemHealth
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
| where LogicalServerName =~ srv
//| where AppName =~ app
| where error_id in (1101, 1105, 9001, 9002)
//| where message contains "error"
| project originalEventTimestamp, NodeName, AppName, error_id, message
| order by originalEventTimestamp asc
| limit 1000
```

Internal Doc Reference

- Temp DB - Resolve tempdb related errors and exceptions
- Temp DB - Tempdb size

How good have you found this content?