MonWiQueryParamData (Compile telemetry)

Last updated by | Francisco O | Jan 17, 2023 at 6:25 AM PST

Contents

Friday, November 10, 2017

6:31 PM

Type of data

One row per each query compile/recompile, containing query/plan hashes and compile stats

Included info

- Compile CPU time (in microseconds)
- Compile duration (in microseconds)
- Info whether plan is cached
- Info whether it is a recompile
- Compile code (Success for compile, specific reason for recompile)
- Info whether query can be parameterized with FORCED parameterization
- Number of parameterized values in the query
- Info whether query has literals

Notable columns to summarize by

- originalEventTimestamp/TIMESTAMP
- Compile_code
- Query_param_type - parameterization type of the query (0 - NONE, 1 - USER, 2 - SIMPLE, 3 - FORCED)
- Query_hash - represents the hash value of the query shape
- Query_plan_hash - represents the hash value of the plan shape

- Statement_sql_hash - represents the hash value of the query text
- Template_hash - if query is parameterizable, represents the templatized query text with FORCED parameterization algorithm (otherwise it is 0x00)

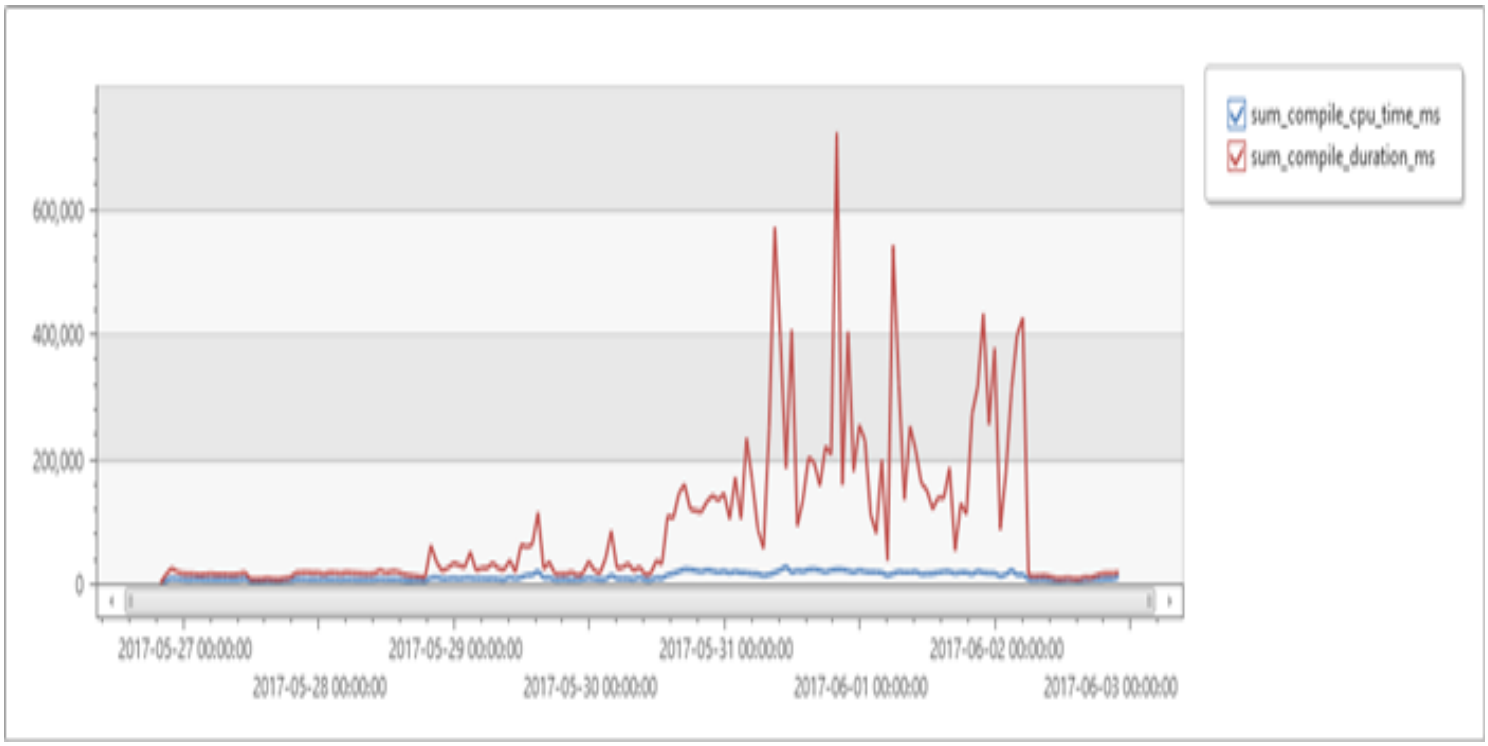
Note: This table contains compiles/recompiles for queries that are supported by QDS, independently from QDS (even if it is turned off or in read-only mode). There is a relatively small set of queries which by design is not tracked in QDS.

Sample queries

Get the total compile CPU time and compile duration of all queries over 1-hour time intervals

https://sqlazrwcus.kusto.windows.net:443/sqlazure1 [Run in Kusto.Explorer] [Run in Kusto.Explorer on SAW] [Run in Kusto.WebExplorer]

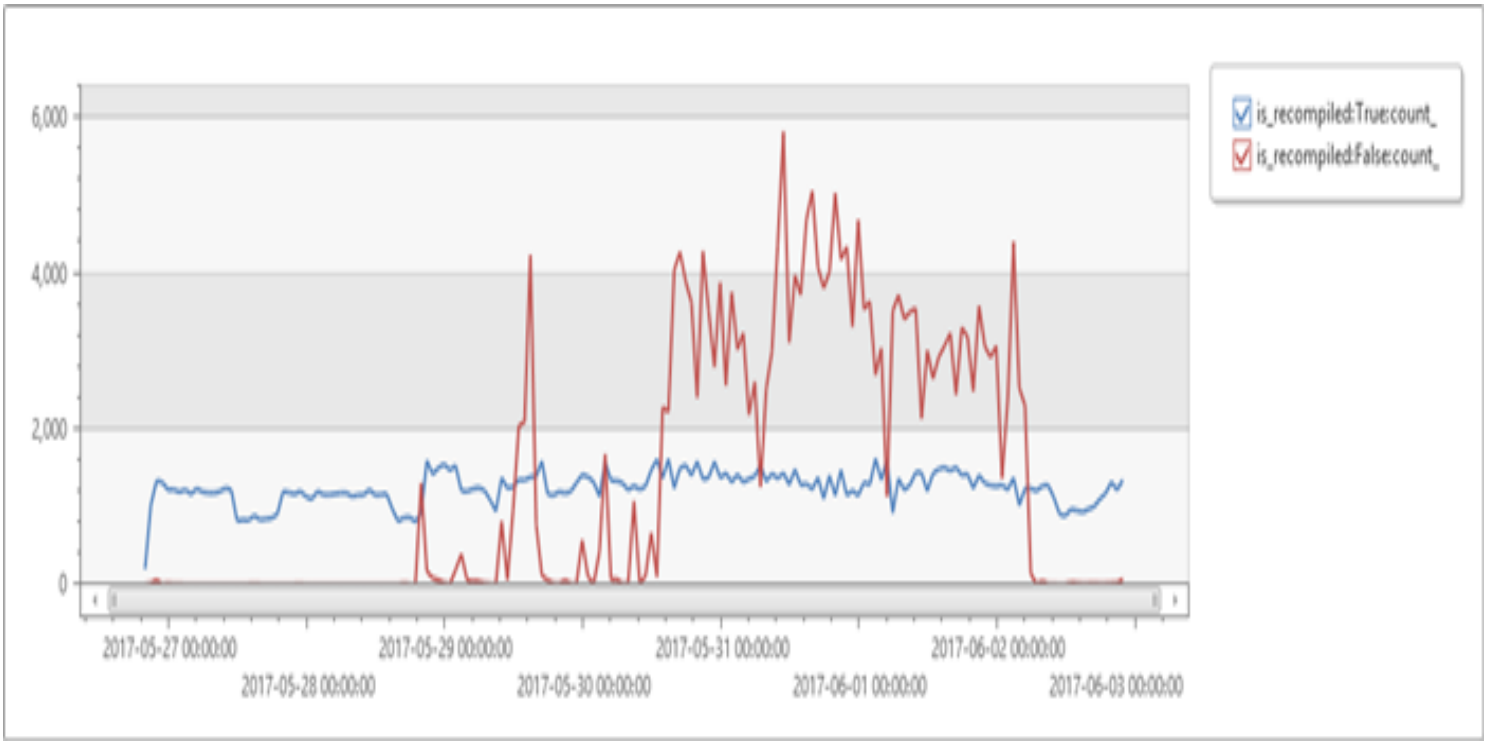
```
MonWiQueryParamData | where LogicalServerName == "v0lmyvu2oy" and logical_database_name == "hosting" | extend compile_cpu_time_ms = 1.0 * compile_cpu_time / 1000 | extend compile_duration_ms = 1.0 * compile_duration / 1000 | summarize sum(compile_cpu_time_ms), sum(compile_duration_ms) by bin(originalEventTimestamp, 1h) | render timechart
```



Get the number of compiles and recompiles over time

https://sqlazrwcus.kusto.windows.net:443/sqlazure1 [Run in Kusto.Explorer] [Run in Kusto.Explorer on SAW] [Run in Kusto.WebExplorer]

```
MonWiQueryParamData | where LogicalServerName == "v0lmyvu2oy" and logical_database_name == "hosting" | summarize count() by is_recompiled, bin(originalEventTimestamp, 1h) | render timechart
```



Get the number of compiles/recompiles by recompile reason (Success means a compile)

https://sqlazrwcus.kusto.windows.net:443/sqlazure1 [Run in Kusto.Explorer] [Run in Kusto.Explorer on SAW] [Run in Kusto.WebExplorer]

```
MonWiQueryParamData | where LogicalServerName == "v0lmyvu2oy" and logical_database_name == "hosting" | summarize count() by compile_code
```

compile_code	count_
Success	231108
StatsChanged	212392
SchemaChanged	222

Figure out if a database has an issue with excessive number of compiles or excessive number of non-parameterized queries

https://sqlazrwcus.kusto.windows.net:443/sqlazure1 [Run in Kusto.Explorer] [Run in Kusto.Explorer on SAW] [Run in Kusto.WebExplorer]

```
MonWiQueryParamData | where LogicalServerName == "v0lmyvu2oy" and logical_database_name == "hosting" | where originalEventTimestamp > now(-7d) | summarize count(), dcount(statement_sql_hash), dcount(query_hash), dcount(query_plan_hash), avg(compile_cpu_time), avg(compile_duration) | extend count_compiles = count_ | extend distinct_query_texts = dcount_statement_sql_hash | extend distinct_query_shapes = dcount_query_hash | extend distinct_plan_shapes = dcount_query_plan_hash | extend avg_compile_cpu_time_ms = 1.0 * avg_compile_cpu_time / 1000 | extend avg_compile_duration_ms = 1.0 *
```

```
avg_compile_duration / 1000 | project count_compiles, distinct_query_texts, distinct_query_shapes,
distinct_plan_shapes, avg_compile_cpu_time_ms, avg_compile_duration_ms
```

count_compiles	distinct_query_texts	distinct_query_shapes	distinct_plan_shapes	avg_compile_cpu_time
440860	1491	961	960	5.6153567753

Explanation of the results:

- A high disproportion between the count of compiles and count of distinct query texts indicates an issue with excessive number of compiles.
- A high disproportion between the count of distinct query texts and distinct query shapes indicates an issue with excessive number of non-parameterized queries.
- A high disproportion between average compile duration and average compile CPU time indicates that a lot of time on compilation is spent waiting.

Top 10 query shapes by the total number of compiles/recompiles

https://sqlazrwcus.kusto.windows.net:443/sqlazure1 [Run in Kusto.Explorer] [Run in Kusto.Explorer on SAW] [Run in Kusto.WebExplorer]

```
MonWiQueryParamData | where LogicalServerName == "v0lmyvu2oy" and logical_database_name == "hosting"
| where originalEventTimestamp > now(-7d) | summarize count(), dcount(statement_sql_hash),
dcount(query_plan_hash), avg(compile_cpu_time), avg(compile_duration) by query_hash | extend
count_compiles = count_ | extend distinct_query_texts = dcount_statement_sql_hash | extend
distinct_plan_shapes = dcount_query_plan_hash | extend avg_compile_cpu_time_ms = 1.0 *
avg_compile_cpu_time / 1000 | extend avg_compile_duration_ms = 1.0 * avg_compile_duration / 1000 | project
query_hash, count_compiles, distinct_query_texts, distinct_plan_shapes, avg_compile_cpu_time_ms,
avg_compile_duration_ms | top 10 by count_compiles
```

query_hash	count_compiles	distinct_query_texts	distinct_plan_shapes	avg_compile_cpu_tim
0xCBB2928B0D2EDE09	34016	1	2	6.202227569
0x09746E9B363093BB	32123	1	1	7.950713165
0xE96578D5E5942183	29984	1	1	4.370890508
0xBB14613E5BFBC808	22778	3	1	5.584911932
0x38E9ACBC20947AFC	16735	1	2	3.184738392
0x18D6E939A1AB8791	10003	4	1	0.861745576
0xDC88329FAE9A029E	9924	1	1	6.469189439
0x0A8A037836361BE3	9923	1	1	4.546215559
0xF0A31446339EA78A	9234	3	2	14.21206681
0x8983744E373F20F8	4370	1	1	5.289574141

Check CPU and percentage of compiles by group before and after parametrization change is made

Internal Reference: IcM 342744819

```
let analyzed_period = 2h;
let change_timestamp = datetime(2023-01-08 09:45:00);
let nodeName = "_DB_61";
let appName = "f8840c28ddce";
let total_compile_cpu_ms_before = toscalar (
MonWiQueryParamData
| where TIMESTAMP > change_timestamp-analyzed_period and TIMESTAMP <= change_timestamp
| where NodeName == nodeName
| where  AppName =~  appName
| project TIMESTAMP, compile_cpu_time
| summarize compile_cpu_ms = sum(compile_cpu_time)/1000);
let total_compile_cpu_ms_after = toscalar (
MonWiQueryParamData
| where TIMESTAMP > change_timestamp and TIMESTAMP <= change_timestamp+analyzed_period
| where NodeName == nodeName
| where  AppName =~  appName
| project TIMESTAMP, compile_cpu_time
| summarize compile_cpu_ms = sum(compile_cpu_time)/1000);
MonSqlRgHistory
| where  TIMESTAMP > change_timestamp-analyzed_period
| where NodeName == nodeName
| where  AppName =~  appName
| where event == "aggregated_workload_groups_plus_history"
| project TIMESTAMP, delta_total_cpu_usage_ms, group_name
| summarize cpu_usage_before_forced_param = sumif(delta_total_cpu_usage_ms, TIMESTAMP > change_timestamp-analy
usage_after_forced_param = sumif(delta_total_cpu_usage_ms, TIMESTAMP > change_timestamp and TIMESTAMP <= chang
| order by cpu_usage_before_forced_param desc
| take 10
| extend total_compile_cpu_ms_before, total_compile_cpu_ms_after
| extend percentage_compile_before = iif ('UserPrimaryGroup.DBId5' == group_name, total_compile_cpu_ms_before*
| extend percentage_compile_after =  iif ('UserPrimaryGroup.DBId5' == group_name,total_compile_cpu_ms_after*1.
```

group_name	cpu_usage_before_forced_param	usage_after_forced_param	total_compile_cpu_ms_before	total_compile_cpu_ms_after	percentage_compile_before	percentage_compile_after
UserPrimaryGroup.DBId5	107616512	74939778	16972878	639115	15.7716299149335	0.59388191284252
InMemQueryStoreGroupLowPri	4153719	1739482	16972878	639115	-1	-1
PVSCleanerGroup	1625401	1008255	16972878	639115	-1	-1
SloSecSharedInitGroup	802740	713495	16972878	639115	-1	-1
GhostCleanupGroup	627669	234747	16972878	639115	-1	-1
InMemQueryStoreGroup	428031	399997	16972878	639115	-1	-1
InMemBackupGroup	371629	287341	16972878	639115	-1	-1
LazywriterGroup	287229	248228	16972878	639115	-1	-1
internal	229166	246519	16972878	639115	-1	-1
CDCScanGroup	186433	63803	16972878	639115	-1	-1

How good have you found this content?