Query Store queries (Managed Instance)

Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:43 PM PDT

```
//*****************************************************************************************************************
// QDS Query Store
//*****************************************************************************************************************
// QDS.01
// Total QDS resource usage by category over time
// absolute values
MonWiQdsExecStats
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where LogicalServerName =~ "{LogicalServerName}"
| where is_primary == 1
| summarize cpu_time_ms = sum(cpu_time) / 1000.0
   , elapsed_time_ms = sum(elapsed_time) / 1000.0
   , sum(logical_reads), sum(execution_count)
   , Total_Memory_MB=sum(max_query_memory_pages*8/1024.0)
   , total_physical_reads=sum(physical_reads)
   , total_num_physical_io_reads=sum(num_physical_io_reads)
   , total_rowcount=sum(rowcount)
   , total_tempdb_space_used_MB=round(sum(tempdb_space_used)*8.0/1024,1)
   by TIMESTAMP=bin(PreciseTimeStamp, 15min)
| render timechart

// total exec count
MonWiQdsExecStats
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where LogicalServerName =~ "{LogicalServerName}"
| where is_primary == 1
| summarize sum(execution_count) by bin(TIMESTAMP, 15min)
| render timechart

// QDS.02
// All QDS CPU Percent over time, relative exec count over time, relative logical reads over time
// CAUTION:
//      1. This number may not be accurate in the presence of queries longer than 15 minutes
let cpu_cap_in_sec=toscalar(
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| summarize total_cpu_ms=(sum(cpu_time)*1.0/(1000)), total_exec_count=sum(execution_count),
```

```
total_logical_reads=sum(logical_reads) by TIMESTAMP=bin(TIMESTAMP, 15min)
| order by TIMESTAMP asc nulls first
| serialize
| extend PrevTimestamp=prev(TIMESTAMP, 1)
| where isnull(PrevTimestamp) == false
| extend elapsed_second = (TIMESTAMP - PrevTimestamp) / 1s
| extend cpu_percent_used = round((total_cpu_ms*100.0/1000)/(elapsed_second * cpu_cap_in_sec),2)
| extend exec_count_per_cpu_per_sec=total_exec_count/(elapsed_second* cpu_cap_in_sec)
| extend logical_reads_per_cpu_per_sec=total_logical_reads/(elapsed_second* cpu_cap_in_sec)
| project TIMESTAMP, cpu_percent_used, exec_count_per_cpu_per_sec //, logical_reads_per_cpu_per_sec
| render timechart

// QDS.03
// top 10 QDS CPU Percent over time
let cpu_cap_in_sec=toscalar(
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where TIMESTAMP >= datetime({StartTime}) and TIMESTAMP <= datetime({EndTime})
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQdsExecStats
  | where LogicalServerName =~ "{LogicalServerName}"
  | where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
  | where is_primary == 1
  | join kind= inner (
   MonWiQdsExecStats
  | where LogicalServerName =~ "{LogicalServerName}"
  | where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
  | where is_primary == 1
  | summarize sum(cpu_time) by query_hash
  | top  10 by  sum_cpu_time desc
) on query_hash
| summarize total_cpu_ms=(sum(cpu_time)*1.0/(1000)) by TIMESTAMP=bin(TIMESTAMP, 15min), query_hash
| order by query_hash asc, TIMESTAMP asc nulls first
| serialize
| extend PrevTimestamp=prev(TIMESTAMP, 1), Prev_query_hash=prev(query_hash, 1)
| where isnull(PrevTimestamp) == false
| where query_hash == Prev_query_hash
| extend elapsed_second = datetime_diff ('second',  TIMESTAMP, PrevTimestamp)
| project TIMESTAMP,  query_hash , cpu_percent_over_dtu=round((total_cpu_ms*100.0/1000)/(elapsed_second*
cpu_cap_in_sec),1)
| render timechart

// top N (5) CPU queries using nested method
let cpu_cap_in_sec=toscalar(
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
```

```
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQdsExecStats
  | where LogicalServerName =~ "{LogicalServerName}"
  | where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
  | where is_primary == 1
  | top-nested of bin(TIMESTAMP, 15m) by sum(cpu_time), top-nested 5 of query_hash by
cpu_percent_over_total_cpu=round((100.0*(sum(cpu_time)/1000.00/1000.0)/cpu_cap_in_sec/(15*60)),2) desc
  | sort by TIMESTAMP asc nulls last
  | project TIMESTAMP, query_hash, cpu_percent_over_total_cpu
  | render timechart


// QDS.04
// top 10 CPU consuming queries over time (absolute value)
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where is_primary  == 1
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| join kind= leftsemi (
   MonWiQdsExecStats
   | where LogicalServerName =~ "{LogicalServerName}"
   | where is_primary == 1
   | where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
   | extend cpu_time_in_hours=cpu_time/ (1.0*1000*1000*60*60)
   | summarize sum(cpu_time_in_hours) by database_name, query_hash
   | top 10 by sum_cpu_time_in_hours  desc nulls last
) on database_name, query_hash
| summarize total_cpu_ms=round(sum(cpu_time)/1000.0,0) by bin(TIMESTAMP, 15min), query_hash =
strcat(database_name, ".", query_hash)
| render timechart

// Top 10 CPU consuming query by hash
// using top-nested
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| extend db_and_query_hash = strcat(database_name, ".", query_hash)
| top-nested of bin(TIMESTAMP, 15m) by sum(cpu_time), top-nested 5 of db_and_query_hash by
total_cpu_ms=round(sum(cpu_time)/1000.0,1) desc
| sort by TIMESTAMP asc nulls last
| project TIMESTAMP, db_and_query_hash, total_cpu_ms
| render timechart

//QDS.05
// compile CPU percent over total CPU available
let cpu_cap_in_sec=toscalar(
```

```
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQueryParamData
| where LogicalServerName =~ "{LogicalServerName}"
| where AppName == "{AppName}" and NodeName =~ "{NodeName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| extend compile_cpu_time_ms = compile_cpu_time / 1000
| extend compile_duration_ms = compile_duration / 1000
| summarize total_cpu_ms=sum(compile_cpu_time_ms) by TIMESTAMP=bin(TIMESTAMP, 5min)
| order by TIMESTAMP asc nulls first
| serialize
| extend PrevTimestamp=prev(TIMESTAMP, 1)
| where isnull(PrevTimestamp) == false
| extend elapsed_second = datetime_diff ('second',  TIMESTAMP, PrevTimestamp)
| project TIMESTAMP, cpu_percent_over_dtu=round((total_cpu_ms*100.0/1000)/(elapsed_second*
cpu_cap_in_sec),4)
| render timechart

// QDS.06
// Specific Query CPU % and exec count per cpu per sec
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| where database_name =~ "{LogicalDatabaseName}" and query_hash =~ "{query_hash}"
| summarize total_cpu_ms=(sum(cpu_time)*1.0/(1000)), total_exec_count=sum(execution_count),
total_logical_reads=sum(logical_reads),distinct_plan_count=dcount(plan_id),
distinct_query_id_count=dcount(query_id), all_plan_ids=makeset(plan_id), all_query_ids=makeset(query_id)

// QDS.06.A
// Specific Query CPU % and exec count per cpu per sec
let cpu_cap_in_sec=toscalar(
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where TIMESTAMP >= datetime({StartTime}) and TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| where database_name =~ "{LogicalDatabaseName}" and query_hash =~ "{query_hash}"
| summarize total_cpu_ms=(sum(cpu_time)*1.0/(1000)), total_exec_count=sum(execution_count),
total_logical_reads=sum(logical_reads),
```

```
    all_plan_ids=makeset(plan_id), all_query_ids=makeset(query_id)
        by TIMESTAMP=bin(TIMESTAMP, 15min)
| order by TIMESTAMP asc nulls first
| serialize
| extend PrevTimestamp=prev(TIMESTAMP, 1)
| where isnull(PrevTimestamp) == false
| extend elapsed_second = (TIMESTAMP - PrevTimestamp) / 1s
| extend cpu_percent_over_dtu=round((total_cpu_ms*100.0/1000)/(elapsed_second* cpu_cap_in_sec),2)
| extend exec_count_per_cpu_per_sec=total_exec_count/(elapsed_second* cpu_cap_in_sec)
| extend logical_reads_per_cpu_per_sec=total_logical_reads/(elapsed_second* cpu_cap_in_sec)
| project TIMESTAMP, cpu_percent_over_dtu,exec_count_per_cpu_per_sec, all_plan_ids, all_query_ids //,
logical_reads_per_cpu_per_sec
| render timechart

// QDS.07
// individual query CPU consumption by plan hash
// One query can have multiple plans
// CAUTION:  this query won't work if a query_hash has many query plans that are used just once
let cpu_cap_in_sec=toscalar(
MonDmRealTimeResourceStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where replica_type == 0
| top 1 by TIMESTAMP desc
| project cpu_cap_in_sec );
MonWiQdsExecStats
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| where database_name =~ "{LogicalDatabaseName}" and query_hash =~ "{query_hash}"
| summarize total_cpu_ms=(sum(cpu_time)*1.0/(1000)), total_exec_count=sum(execution_count),
total_logical_reads=sum(logical_reads)
        by query_plan_hash, TIMESTAMP=bin(TIMESTAMP, 15min)
| order by query_plan_hash asc nulls first , TIMESTAMP asc nulls first
| serialize
| extend PrevTimestamp=prev(TIMESTAMP, 1), prev_key=prev(query_plan_hash, 1)
| where isnotnull(PrevTimestamp)
| where query_plan_hash == prev_key
| extend elapsed_second = (TIMESTAMP - PrevTimestamp) / 1s
| extend cpu_percent_used = round((total_cpu_ms*100.0/1000)/(elapsed_second* cpu_cap_in_sec),2)
| extend exec_count_per_cpu_per_sec=total_exec_count/(elapsed_second* cpu_cap_in_sec)
| extend logical_reads_per_cpu_per_sec=total_logical_reads/(elapsed_second* cpu_cap_in_sec)
| project TIMESTAMP, query_plan_hash, cpu_percent_used, exec_count_per_cpu_per_sec  //,
logical_reads_per_cpu_per_sec
| render timechart

// QDS.08
// Specific query absolute value usage
MonWiQdsExecStats
```

```
| where LogicalServerName =~ "{LogicalServerName}"
| where  TIMESTAMP >= datetime({StartTime}) and  TIMESTAMP <= datetime({EndTime})
| where is_primary == 1
| where database_name =~ "{LogicalDatabaseName}" and query_hash =~ "{query_hash}"
//| where  query_id == 438581  // there can be many query_id for a query_hash if not parameterized
| summarize  distinct_plan_id_count=dcount( plan_id)
    , distinct_plan_count = dcount(query_plan_hash)
    , total_execution_count = sum(execution_count)
    , min_elapsed_ms = round(min(min_elapsed_time)/1000.0,1)
    , avg_elapsed_time_ms = round(sum(elapsed_time) / ( 1000. * sum(execution_count)),1)
    , max_elapsed_ms = round(max(max_elapsed_time)/1000.0,1)
    , total_elapsed_time_Ms= round(sum(elapsed_time)/1000.0,1)
    , min_cpu_ms=round(min(min_cpu_time)/1000.0,1)
    , avg_cpu_ms = round(sum( cpu_time )/1000.0 / sum( execution_count),2)
    , max_cpu_ms= round(max(max_cpu_time)/1000.0,1)
    , total_cpu_ms=round(sum(cpu_time)/1000.0,1)
    , min_logical_reads=min(min_logical_reads)
    , avg_logical_reads =  sum(logical_reads) / sum(execution_count)
    , max_logical_reads=max(max_logical_reads)
    , total_logical_reads=sum(logical_reads)
    , avg_rowcount = sum( rowcount) / sum( execution_count )
    ,min_num_physical_io_reads = min( min_num_physical_io_reads)
    ,avg_num_physical_io_reads = sum( num_physical_io_reads) / sum(execution_count )
    ,max_num_physical_io_reads = max( max_num_physical_io_reads)
    ,total_num_physical_io_reads = sum(num_physical_io_reads)
    ,min_physical_reads = min(min_physical_reads)
    , avg_physical_reads = sum( physical_reads) / sum(execution_count )
    ,max_physical_reads = max(max_physical_reads)
    , total_physical_reads=sum(physical_reads)
    by key="" , bin(TIMESTAMP, 1h)
    //by key = strcat( query_id, ' ', plan_id), bin(TIMESTAMP, 1h)
   // by key = strcat( query_hash,  ' ', query_plan_hash), bin(TIMESTAMP, 1h)
    //by key = strcat( query_id, ''), bin(TIMESTAMP, 1h)
    //| project TIMESTAMP , key,  avg_elapsed_time_ms , avg_cpu_ms, avg_logical_reads, avg_physical_reads,
avg_reads
    //| project TIMESTAMP , key,   max_logical_reads,  min_logical_reads, avg_logical_reads
| project TIMESTAMP, key, max_cpu_ms,  min_cpu_ms, avg_cpu_ms, total_cpu_ms, total_elapsed_time_Ms,
distinct_plan_count
| render timechart

// QDS.09
// QDS failures
MonQueryStoreFailures
| where LogicalServerName =~ "{LogicalServerName}" and logical_database_name =~ "{LogicalDatabaseName}"
| summarize count(), min(TIMESTAMP), max(TIMESTAMP) by query_id, plan_id, event, error_number
```

How good have you found this content?