Stored Procedure to Get Expensive Queries

https://gallery.technet.microsoft.com/scriptcenter/Stored-Procedure-to-Get-b5132c3d

As a DBA we often need to get information on various Queries, One of them is how expensive it is.

A query can be judged as expensive on various criteria's like Long Running, Memory Utilization, CPU utilization etc.

We came up with this SP which gives you Expensive Queries based on criteria's like Duration, Memory, CPU, Read etc.

Double Click and execute the attached SP on master database.

SP to get Expensive Queries.txt

/* Example-1: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 0 */

/* Example-2: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 1 */

/* Example-3: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 2 */

/* Example-4: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 3 */

/* Example-5: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 4 */

/* Example-6: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 5 */

/* Example-7: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 6 */

/* Example-8: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 7 */

--Main SP:

```
USE [master]
GO

IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[usp_GetExpensiveQueries]') AND type in (N'P',
N'PC'))
DROP PROCEDURE [dbo].[usp_GetExpensiveQueries]
GO

USE [master]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[usp_GetExpensiveQueries]
@Limit AS INT,
@Database_Name AS VARCHAR(255),
@Expense_Counter INT
AS

/*Variables Used

1. @Limit -- No of Recoreds to be Retrieved in Int
2. @Database_Name -- Database Name for which the Expense queries needs to be Retrieved in Varchar
3. @Expense_Counter -- Criteria on Which the Query Expense needs to judged

Refer Below for Expense Counter End Variables Used*/

/*Expense Counter

DurTimeAvgMin 0
CPUTimeAvgMin 1
TotalCPUTime 2
TotalDurTime 3
NoPhysicalReads 4
AvgNoPhyscialReads 5
NoLogicalReads 6
AvgNoLogicalReads 7

*/

/* Example-1: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 0 */
/* Example-2: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 1 */
/* Example-3: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 2 */
/* Example-4: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 3 */
/* Example-5: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 4 */
/* Example-6: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 5 */
/* Example-7: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 6 */
/* Example-8: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 7 */


SELECT Database_name,QueryText,ProcBatTest,PlanGenerationNumber,ExecutionCount,DurTimeAvgMin,
CPUTimeAvgMin,TotalCPUTime,TotalDurTime,NoPhysicalReads,AvgNoPhyscialReads,NoLogicalReads,AvgNoLogicalReads
FROM
(
SELECT TOP (@Limit) DB_NAME(CONVERT (INT, epa.value)) AS [Database_Name],
```

```
SUBSTRING(est.text, (eqs.statement_start_offset/2)+1,
        ((CASE eqs.statement_end_offset
         WHEN -1 THEN DATALENGTH(est.text)
         ELSE eqs.statement_end_offset
         END - eqs.statement_start_offset)/2) + 1) AS QueryText,
est.text AS ProcBatTest,
eqs.plan_generation_num AS PlanGenerationNumber,
eqs.execution_count AS ExecutionCount,
(eqs.total_worker_time/1000) AS TotalCPUTime,
(((eqs.total_worker_time/1000)/eqs.execution_count)/3600) AS CPUTimeAvgMin,
(eqs.total_elapsed_time/1000) AS TotalDurTime,
(((eqs.total_elapsed_time/1000)/eqs.execution_count)/3600) AS DurTimeAvgMin,
eqs.total_physical_reads AS NoPhysicalReads,
(eqs.total_physical_reads/eqs.execution_count) AS AvgNoPhyscialReads,
eqs.total_logical_reads AS NoLogicalReads,
(eqs.total_logical_reads/eqs.execution_count) AS AvgNoLogicalReads,
eqs.last_execution_time AS LastExecutionTime
FROM SYS.DM_EXEC_QUERY_STATS eqs
CROSS APPLY SYS.DM_EXEC_SQL_TEXT(sql_handle) est
CROSS APPLY SYS.DM_EXEC_QUERY_PLAN(plan_handle) eqp
CROSS APPLY SYS.DM_EXEC_PLAN_ATTRIBUTES(eqs.plan_handle) epa
WHERE attribute = 'dbid'
AND DB_NAME(CONVERT (INT, epa.value)) = @Database_Name) x
--and qs.last_execution_time > '2011-08-09 17:29:33.750'
--If we want to get queries executed greater than some time
--and (((qs.total_elapsed_time/1000)/qs.execution_count)/3600) >= 2
ORDER BY
--Seems to be Problem with Order By working on the same
--Order By Fixed
CASE
WHEN @Expense_Counter = 0 THEN DurTimeAvgMin
WHEN @Expense_Counter = 1 THEN CPUTimeAvgMin
WHEN @Expense_Counter = 2 THEN TotalCPUTime
WHEN @Expense_Counter = 3 THEN TotalDurTime
WHEN @Expense_Counter = 4 THEN NoPhysicalReads
WHEN @Expense_Counter = 5 THEN AvgNoPhyscialReads
WHEN @Expense_Counter = 6 THEN NoLogicalReads
WHEN @Expense_Counter = 7 THEN AvgNoLogicalReads
END DESC
GO
```

--Execute the below to get output:

```
/* Example-1: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 0 */
/* Example-2: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 1 */
/* Example-3: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 2 */
/* Example-4: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 3 */
/* Example-5: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 4 */
/* Example-6: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 5 */
/* Example-7: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 6 */
/* Example-8: [usp_GetExpensiveQueries] @Limit =10, @Database_Name = 'sample1',@Expense_Counter = 7 */
```