# **HIGH CPU SQL SERVER SOLUTION Using** statistics and Indexs

Performance Tuning pada database terus dikembangkan agar dapat memudahkan DBA untuk lebih cepat mengidentifikasi *bottleneck*, menargetkan operasi yang tidak sesuai dengan melihat hasil dari *query execution plan*, dan menghilangkan dugaan-dugaan yang tidak pasti.

1. Berikut untuk melihat queries running dan seberapa banyak CPU yang digunakan pada database: (klik new query pada database yang akan dinanalisa)

Graphical user interface, application

Description automatically generated

SELECT TOP 100 s.session\_id,

r.status,

r.cpu\_time,

r.logical\_reads,

r.reads,

r.writes,

r.total\_elapsed\_time / (1000 \* 60) 'Elaps M',

SUBSTRING(st.TEXT, (r.statement\_start\_offset / 2) + 1,

((CASE r.statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.TEXT)

ELSE r.statement\_end\_offset

END - r.statement\_start\_offset) / 2) + 1) AS statement\_text,

COALESCE(QUOTENAME(DB\_NAME(st.dbid)) + N'.' + QUOTENAME(OBJECT\_SCHEMA\_NAME(st.objectid, st.dbid))

+ N'.' + QUOTENAME(OBJECT\_NAME(st.objectid, st.dbid)), '') AS command\_text,

r.command,

s.login\_name,

s.host\_name,

s.program\_name,

s.last\_request\_end\_time,

s.login\_time,

r.open\_transaction\_count

FROM sys.dm\_exec\_sessions AS s

JOIN sys.dm\_exec\_requests AS r ON r.session\_id = s.session\_id CROSS APPLY sys.Dm\_exec\_sql\_text(r.sql\_handle) AS st

WHERE r.session\_id != @@SPID

ORDER BY r.cpu\_time DESC

Jika tidak ada yang berjalan, kita bisa melihat history yang sudah ada seberapa banyak CPU queries yang berjalan :

Graphical user interface, application

Description automatically generated

SELECT TOP 10 st.text AS batch\_text,

SUBSTRING(st.TEXT, (qs.statement\_start\_offset / 2) + 1, ((CASE qs.statement\_end\_offset WHEN - 1 THEN DATALENGTH(st.TEXT) ELSE qs.statement\_end\_offset END - qs.statement\_start\_offset) / 2) + 1) AS statement\_text,

(qs.total\_worker\_time / 1000) / qs.execution\_count AS avg\_cpu\_time\_ms,

(qs.total\_elapsed\_time / 1000) / qs.execution\_count AS avg\_elapsed\_time\_ms,

qs.total\_logical\_reads / qs.execution\_count AS avg\_logical\_reads,

(qs.total\_worker\_time / 1000) AS cumulative\_cpu\_time\_all\_executions\_ms,

(qs.total\_elapsed\_time / 1000) AS cumulative\_elapsed\_time\_all\_executions\_ms

FROM sys.dm\_exec\_query\_stats qs

CROSS APPLY sys.dm\_exec\_sql\_text(sql\_handle) st

ORDER BY(qs.total\_worker\_time / qs.execution\_count) DESC

1. Update statistics

exec sp\_updatestats (jalankan pada database yang diinginkan ,execute)

pengertian statistics :

The sp\_updatestats system stored procedure runs UPDATE STATISTICS against all user-defined and internal tables in the current database. For regular maintenance, ensure that regularly schedule maintenance is keeping statistics up to date. Use solutions such as [**Adaptive Index Defrag**](https://github.com/Microsoft/tigertoolbox/tree/master/AdaptiveIndexDefrag) to automatically manage index defragmentation and statistics updates for one or more databases. This procedure automatically chooses whether to rebuild or reorganize an index according to its fragmentation level, among other parameters, and update statistics with a linear threshold.

1. Add missing indexes

Pertama kita dapat mengecek ,pada queries database yang setidaknya bisa kita masukan missing index , dengan klik pada “query\_plan”.

Graphical user interface, application

Description automatically generated

-- Captures the Total CPU time spent by a query along with the query plan and total executions

SELECT

qs\_cpu.total\_worker\_time / 1000 AS total\_cpu\_time\_ms,

q.[text],

p.query\_plan,

qs\_cpu.execution\_count,

q.dbid,

q.objectid,

q.encrypted AS text\_encrypted

FROM

(SELECT TOP 500 qs.plan\_handle,

qs.total\_worker\_time,

qs.execution\_count FROM sys.dm\_exec\_query\_stats qs ORDER BY qs.total\_worker\_time DESC) AS qs\_cpu

CROSS APPLY sys.dm\_exec\_sql\_text(plan\_handle) AS q

CROSS APPLY sys.dm\_exec\_query\_plan(plan\_handle) p

WHERE p.query\_plan.exist('declare namespace

qplan = "http://schemas.microsoft.com/sqlserver/2004/07/showplan";

//qplan:MissingIndexes')=1

Membuat missing index secara automatis dengan queries berikut

Graphical user interface, text, application

Description automatically generated

SELECT CONVERT(VARCHAR(30), GETDATE(), 126) AS runtime,

mig.index\_group\_handle,

mid.index\_handle,

CONVERT(DECIMAL(28, 1), migs.avg\_total\_user\_cost \* migs.avg\_user\_impact \* (migs.user\_seeks + migs.user\_scans)) AS improvement\_measure,

'CREATE INDEX missing\_index\_' + CONVERT(VARCHAR, mig.index\_group\_handle) + '\_' + CONVERT(VARCHAR, mid.index\_handle) + ' ON ' + mid.statement + ' (' + ISNULL(mid.equality\_columns,

'') + CASE WHEN mid.equality\_columns IS NOT NULL

AND mid.inequality\_columns IS NOT NULL THEN ','

ELSE ''

END + ISNULL(mid.inequality\_columns,

'') + ')' + ISNULL(' INCLUDE (' + mid.included\_columns + ')',

'') AS create\_index\_statement,

migs.\*,

mid.database\_id,

mid.[object\_id]

FROM sys.dm\_db\_missing\_index\_groups mig

INNER JOIN sys.dm\_db\_missing\_index\_group\_stats migs ON migs.group\_handle = mig.index\_group\_handle

INNER JOIN sys.dm\_db\_missing\_index\_details mid ON mig.index\_handle = mid.index\_handle

WHERE CONVERT (DECIMAL (28, 1),

migs.avg\_total\_user\_cost \* migs.avg\_user\_impact \* (migs.user\_seeks + migs.user\_scans)) > 10

ORDER BY migs.avg\_total\_user\_cost \* migs.avg\_user\_impact \* (migs.user\_seeks + migs.user\_scans) DESC

Pada gambar diatas, dapat dilihat create\_index\_statement , copy all dan jalankan pada database tersebut.

Graphical user interface

Description automatically generated with medium confidence