Resources Governor Explained

Choirul Amri
http://choirulamri.org

SQL Server User Group Indonesia
Session Objectives And Agenda

• What Resource Governor is
  – Scenarios covered
  – Limitations in scope and scenarios

• Concepts and details

• CPU Demo

• Feedback, Q&A
Why Resource Governor

- You want to control resources usage
- Prevent a single application from overrunning the system
  - Control the apps regardless off their code
- Balance resources usage/workload among applications
  - Allocate more or less resources to specific application
Life without Resources Governor

- Kill runaway SPIDs
- Setup instances and affinity settings
- Manually schedule jobs so they don’t overlap
- Create separate VMs so you can allocate memory and CPU
- SET QUERY_GOVERNOR_COST_LIMIT
Resource Governor Limitations

- **Database Engine only**
  - Analysis Services, Reporting Services, etc. are covered as “regular” applications to SQL Server database engine

- **Controls for CPU bandwidth and Memory**
  - No I/O governor

- **Single instance only**
  - Each instance controlled individually
  - Can be combined with Windows Server Resource Manager (WSRM)
Concepts To Understand

• **Resource pools**
  – A possible subset of server resources (CPU, memory)
  – A pool of CPU and memory resource

• **Workload groups**
  – A way to subdivide resource pools (sales, HR)
  – Slices of resource pools for specific scope: apps/db/user

• **Classification function**
  – How the resources a grouped and mapped to workload group
  – It’s a custom function
Resource Pool – Workload relationship
Resource Pools

• A pool of CPU and memory resource
• There are 2 defaults:
  – Internal
• Internal SQL operation — Default
• Any
• processes doesn’t match with classifier

• RP settings are applicable to each CPU/memory available for use by SQL Server
Resource Pool Settings - CPU

• **Minimum CPU%**
  – MIN_CPU_PERCENT
  – Minimum guaranteed amount of CPU time
    • For all requests in pool when there is contention
    • 0 (default) \( \leq \) MIN_CPU_PERCENT \( \leq \) 100
    • Sum of this value for all RP must be \( \leq \)100

• **Maximum CPU %**
  – MAX_CPU_PERCENT
  – Maximum average amount of CPU time
    • For all request in pool when there is contention
    • MIN_CPU_PERCENT \(<\) MAX_CPU_PERCENT \(\leq\) 100 (Default)
Resource Pool Settings - Memory

• **Minimum Memory %**
  – MIN_MEMORY_PERCENT
  – Minimum memory dedicated to this RP (nor shared with RPs)

• 0 (Default) <= MIN_MEMORY_PERCENT <= 100

• Sum of this value for all resources pools must be <= 100

• **Maximum Memory %**
  – MAX_MEMORY_PERCENT
  – The maximum amount of memory that can be used by the resource pool
    • MIN_MEMORY_PERCENT < MAX_MEMORY_PERCENT <= 100
**Resource Pool Calculation**

<table>
<thead>
<tr>
<th>Pool name</th>
<th>Min CPU %</th>
<th>Max CPU %</th>
<th>Calculated Effective Max %</th>
<th>Calculated Shared %</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>20</td>
<td>100</td>
<td>55</td>
<td>35</td>
</tr>
<tr>
<td>LowPool</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>HighPool</td>
<td>25</td>
<td>80</td>
<td>60</td>
<td>35</td>
</tr>
</tbody>
</table>

60 = MIN (80, (100-20-20))

<table>
<thead>
<tr>
<th>Pool name</th>
<th>Min CPU %</th>
<th>Max CPU %</th>
<th>Calculated Effective Max %</th>
<th>Calculated Shared %</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>20</td>
<td>100</td>
<td>55</td>
<td>35</td>
</tr>
<tr>
<td>LowPool</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>HighPool</td>
<td>25</td>
<td>80</td>
<td>60</td>
<td>35</td>
</tr>
</tbody>
</table>

30 = 50 - 20
Workload Group

• An RP can be associated to multiple Workload Groups
  – A WG can only be associated to single RP

• Can be set to low, medium, or high importance

• It’s a balance ratio
  – Low:Medium:High = 1:3:9

• Associate incoming requests to RP
Workload Group Settings

- **REQUEST_MAX_MEMORY_GRANT_PERCENT**
  - Percent memory RELATIVE to RP
  - 0 to 100 (default is 25)
  - If larger than 50, big query run one at a time

- **REQUEST_MAX_CPU_TIME_SEC**
  - Maximum amount CPU time
  - If exceeded, it doesn’t stop instead generates trace event
  - Default value: 0 (unlimited)
Workload Group Settings

• **GROUP_MAX_REQUESTS**
  – Max number of simultaneous requests that are allowed to execute in a WG
  – Default: 0 (unlimited)

• **IMPORTANCE**
  – Local to RP
  – LOW, MEDIUM, HIGH. Default: MEDIUM
Classifier Function

• A user defined function located in the master database

• Executed for each incoming session
  – Return type: the name of Workload Group — It maps incoming request to particular WG

• Context criteria example:
  – USER_NAME()
  – APP_NAME(), HOST_NAME()
  – IS_MEMBER(), GETDATE()
Classifier Function Example

CREATE FUNCTION dbo.fn_ClassifyApps() RETURNS sysname

Create a classifier function

WITH SCHEMABINDING
AS
BEGIN
    DECLARE @ret sysname
    IF (APP_NAME() LIKE '%Low Importance Application%')
      SET @ret='Low Importance Group'
    RETURN @ret
END
GO
ALTER RESOURCE GOVERNOR
  WITH (CLASSIFIER_FUNCTION = dbo.fn_ClassifyApps)
ALTER RESOURCE GOVERNOR RECONFIGURE
Monitoring

- **Performance counters**
  - Instance per Pool: *SQLServer: Resource Pool Stats*
  - Instance per Group: *SQLServer: Workload Group Stats*

- **DMVs:**
  - 3 new: 1 for groups, 1 for pools, 1 for configuration
  - Existing DMVs to include group/pool ID (e.g. `sys.dm_exec_sessions`)

- **Catalog views:**
  - 3 new: metadata about configuration
  - Can be used to transfer configuration to another server
Demo

Resource Governor
Q/A

- Thank you 😊
- Download PPT and sample scripts from my blog
  – http://choirulamri.org