Lesson Objectives

- In this Lesson you will learn the following:
  - SQL Server Operating System (SQLOS)
  - SQL Server Memory Management
  - SQL Server Disk I/O
  - SQL Server Process Scheduling
  - SQL Server Wait Statistics
  - SQL Server Resource Governor

SQL Server Operating System

Leveraging Resource Governor in SQL Server

SQL Server will execute queries without bias, first come.. first serve. Consolidation, Hosting, and Scenarios where *Mixed Workloads* can threaten **‘Predictable Performance’**

Introduction to Resource Governor

- Introduced in SQL 2008 to promote Predictable Performance
- Available only on the Enterprise, Developer, and Evaluation editions of SQL Server and only applicable to the database engine
- In SQL Server 2014 Resource Governor allows the governance of the following resources:
  - CPU
  - Memory
  - I/O

Resource Governor Scenarios

Mixed Workload Types

- A resource intensive report can take up most or all of the server resources

Hosting Companies

- A costumer consumes high CPU percentage, leading to complaint from other customers

Consolidation

- Sharing resources

Resource Governor – Feature Overview

- Resource Governor was first introduced
- CPU and Memory Governance
- CPU_CAP_PERCENT
- Affinity and NUMA
- IO Governance

Feature Goals

- Minimize runaway queries
- Improve the predictable performance of queries across the SQL Server engine

Benefits of Resource Governor

- Limit large resource consumption
  - Expensive reports
  - Run-away queries
  - Ad-hoc queries
- Protect mission critical workloads
- Provide a custom monitoring solution

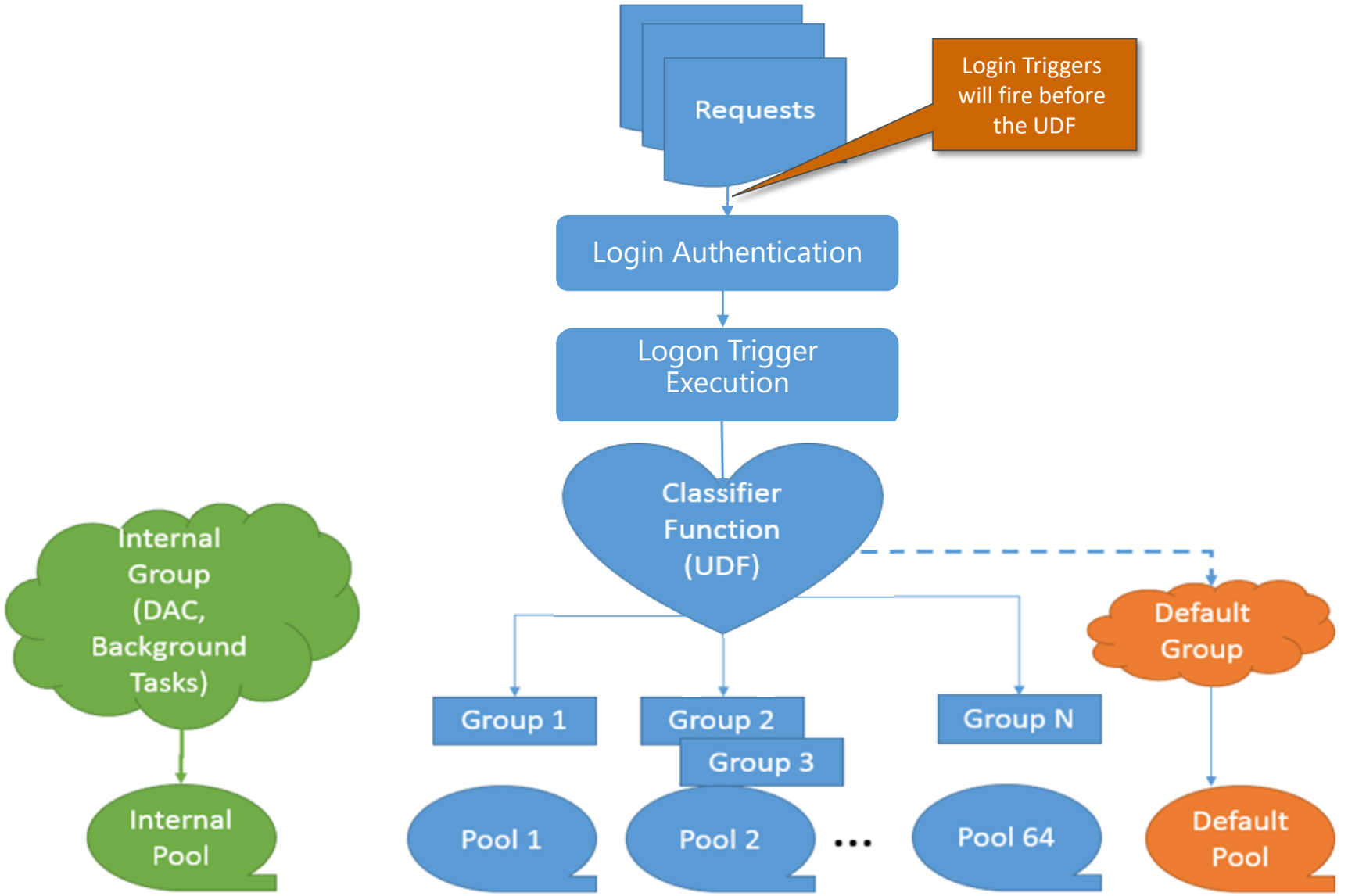
Lesson Knowledge Check

- On which SQL Server Editions is Resource Governor available?
- What SQL Server version introduced the possibility to govern IO?
- True or False: Resource Governor allows for running a large report without interfering performance for other users ?

Leveraging Resource Governor in SQL Server

Resource Governor Architecture

Resource Governor Architecture



Resource Governor Classifier Function

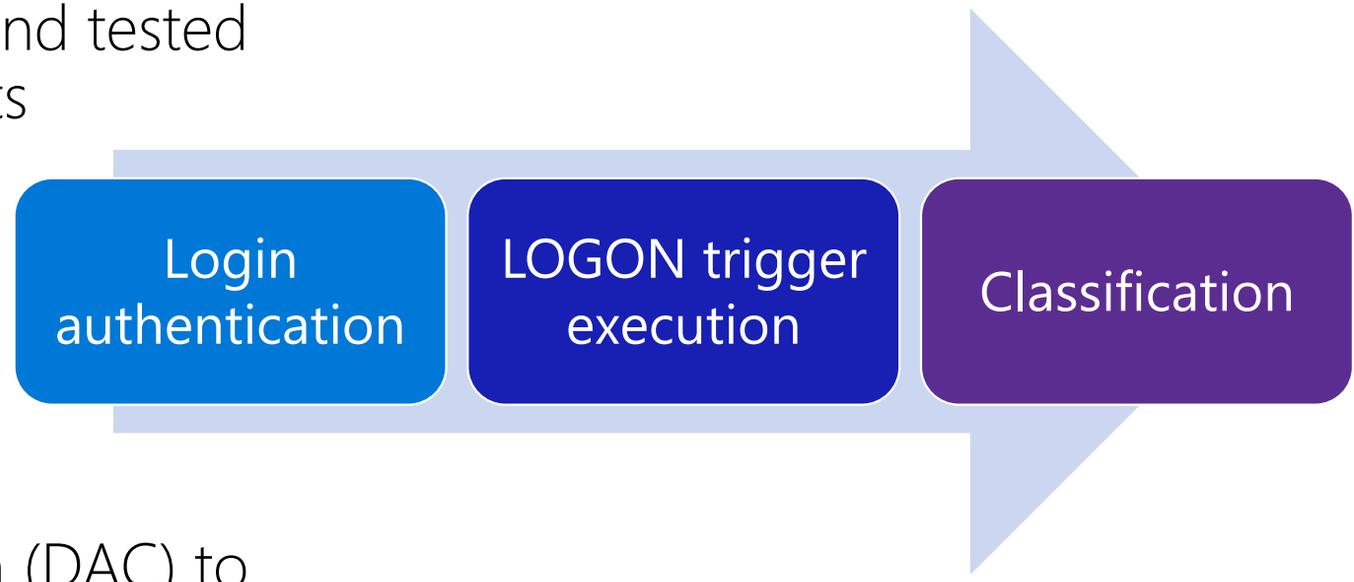
- User Defined Function (UDF)
  - Classifies connections into workgroups
  - Must exist in master database
  - Runs after logon triggers
  - Runs under the internal resource pool

- System functions that can be used:
  - HOST_NAME(), APP_NAME(), SUSER_NAME(), SUSER_SNAME(), IS_SRVROLEMEMBER(), and IS_MEMBER()
  - LOGINPROPERTY() and ConnectionProperty()

- If the Classifier UDF causes an error or does not classify a connection, the default pool is used

Classifier Function Performance Considerations

- Classifier function needs to be optimized and tested before using it to classify incoming requests
- A poorly written function can render the system unusable by causing login timeouts



- Use a Dedicated Administrator Connection (DAC) to troubleshoot a classifier function while Resource Governor is enabled as this connection is not subject to classification
- Leverage the PreConnect:Completed Event Class to monitor the Classifier function overhead

Classifier Function DDL

```
USE [master];
GO
CREATE FUNCTION dbo.fn_classifier()
	RETURNS SYSNAME  -- ← Critical to return the group name with the same
case
	WITH SCHEMABINDING
AS
BEGIN
	IF <condition1>
		RETURN (SELECT N'group1');
	…
	IF <conditionN>
		RETURN (SELECT N'groupN');
	RETURN (SELECT N'default');
END
GO
```

Workload Groups

- Serves as a container for session requests that are similar according to the classification criteria
- Each workload group is assigned to only one resource pool
- A workload group can be dropped or moved to another resource pool while it contains active sessions, but cannot apply the change until the workgroup is empty
- Allows the aggregate monitoring of resource consumption and the application of a uniform policy to all the requests in the group

Predefined Workload Groups

- When SQL Server is installed the following workload groups are defined:
  - **Internal Workload Group** (group_id = 1)
  - **Default Workload Group** (group_id = 2)

- Requests fall into the **default group** when:
  - There are no criteria to classify a request
  - There is an attempt to classify the request into a non-existent group
  - There is an attempt to classify the request into the Internal workload group
  - There is a general classification failure

WORKLOAD GROUP DDL

```
CREATE WORKLOAD GROUP group_name
[ WITH
      ( [ IMPORTANCE = { LOW | MEDIUM | HIGH } ]
        [ [ , ] REQUEST_MAX_MEMORY_GRANT_PERCENT = valu
        [ [ , ] REQUEST_MAX_CPU_TIME_SE
        [ [ , ] REQUEST_MEMORY_GRANT_TIMEOUT
        [ [ , ] MAX_DOP = value ]
        [ [ , ] GROUP_MAX_REQUESTS = v
]
[ USING { pool_name | "default" } ]
[ ; ]
```

Allows the scheduler to balance work per workload

Number of concurrent requests running within the workload group

granted to any query in the workload group

Prevents a parallel plan om exceeding the # of a equest is queued

Number of seconds a query will wait for a memory grant

Polling mechanism fires an event the first time a query exceeds the limit

Resource Pools

- Represents the physical resources on a server
- Pools control MIN and MAX for CPU, Memory, and Disk (SQL 2014+)
  - MINs are defined as non-shared. It does not overlap with other pools, which enables minimum reservation.
  - Sum of MIN values across all Pools cannot exceed 100% of server resources
  - MAX is defined as shared with other pools, which supports maximum possible resource consumption. MAX value can be set anywhere in the range between MIN and 100% inclusive
- SQL Server 2014+ allows for up to 64 resource pools (SQL 2008 allowed 20)
- One or more workload groups can be assigned to a resource pool

Predefined Resource Pools

When SQL Server is installed the following Resource Pools are defined:

- Internal Resource Pool (pool_id =1)

  It is the Pool allocated to SQL Server background process. Any workloads in this pool are considered critical for SQL Server function, and Resource Governor

  Pool cannot be altered in any way

  Resource consumption is not restricted and not subtracted from the overall resource usage

  The internal pool can, and will, pressure other pools even if it means the violation of limits set for the other pools

- Default Resource Pool (pool_id = 2)

  It is the first pre-defined user Resource Pool

  Prior to any configurations, the default Resource Pool contains only the Default Workload Group

  The DEFAULT RESOURCE GROUP can not be created or dropped, but it can be altered. It can contain other user-defined workload groups in addition to the DEFAULT

RESOURCE POOL DDL

```
CREATE RESOURCE POOL pool_name
[ WITH
        (
                                  PU_PERCENT = value ]
                                  MAX_CPU_PERCENT = value ]
                                  CAP_CPU_PERCENT = value ]
                                  AFFINITY {SCHEDULER =
                                 AUTO | ( <scheduler_range_sp  > )
                                  | NUMANODE = ( <NUMA_node_range_spe
                        [ [ , ] MIN_MEMORY_PERCENT = value ]
                        [ [ , ] MAX_MEMORY_PERCENT = valu
                        [ [ , ] MIN_IOPS_PER_VOLUME = valu
                        [ [ , ] MAX_IOPS_PER_VOLUME = valu
        )
] [;]
```

Minimum I/O operations per second (IOPS) per disk volume to reserve for the resource pool

Applies when there is CPU contention and represents the % of scheduler time

Hard cap of CPU scheduler time

Affinitize to one or more

WARNING: Ensures % of memory will be available on a given pool. Memory remains in the pool even when the pool doesn't have any requests. This could cause memory to be unavailable for other pools

Enabling and Disabling Resource Governor

- Resource Governor is enabled after installation
  (User configuration must be completed manually)
- Changes to Resource Governor are not automatically propagated to new or existing connections
- Enabling the Resource Governor requires CONTROL SERVER permission
- To start or apply changes to Resource Governor, run the following:

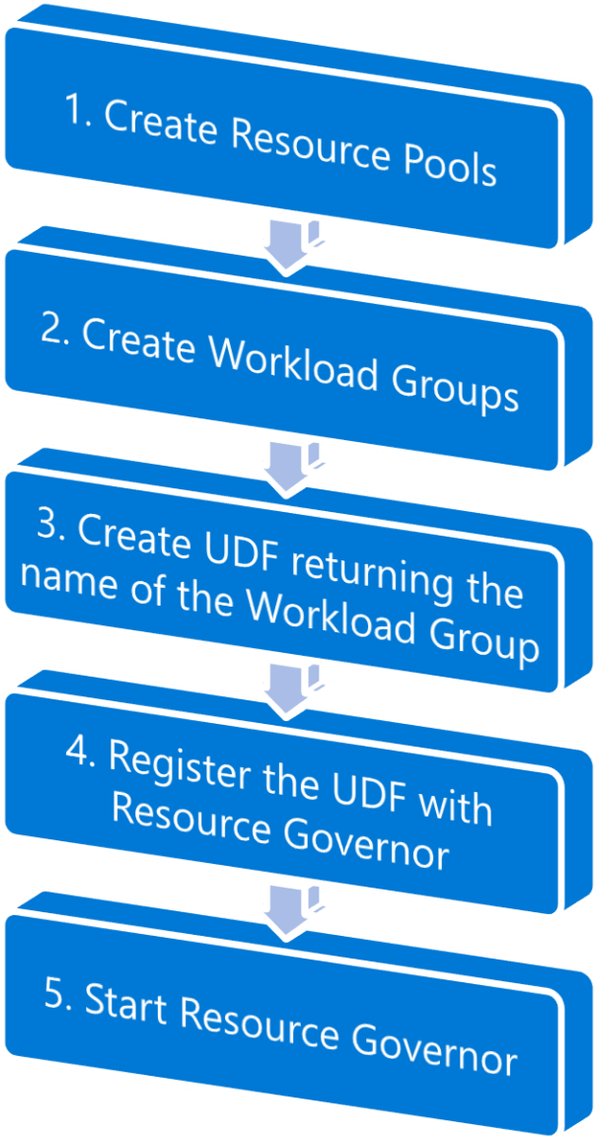
```
ALTER RESOURCE GOVERNOR RECONFIGURE
```

- To disable Resource Governor, run the following:

```
ALTER RESOURCE GOVERNOR DISABLE
```

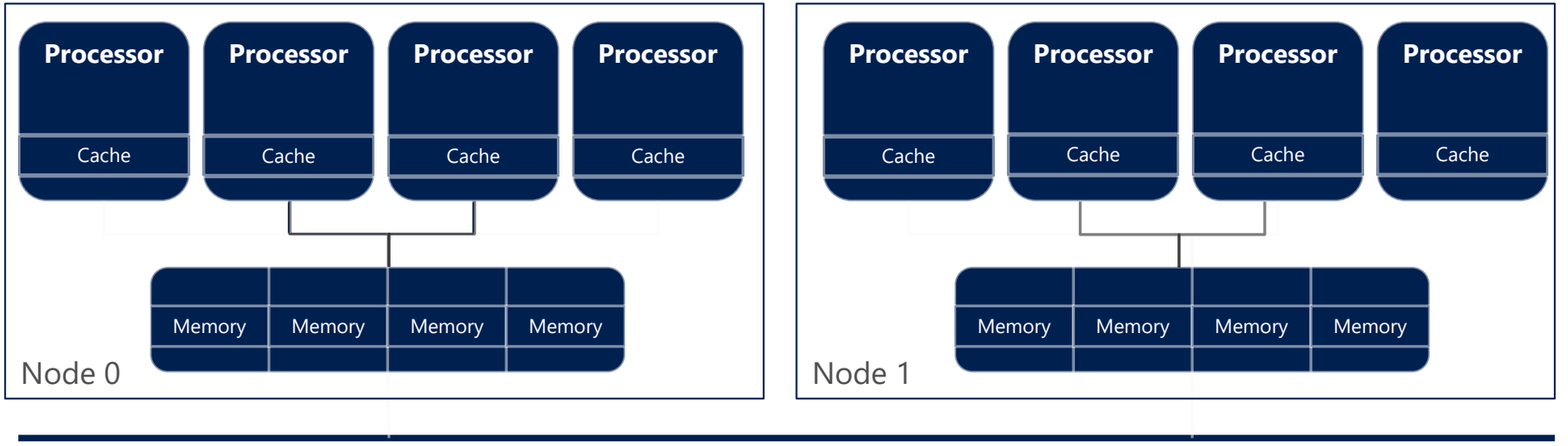
Overall Process for Configuring Resource Governor

1. Create additional and/or alter existing resource pools with the appropriate settings
2. Create additional and/or alter existing workload group(s) with the appropriate settings and assign each workload group to a specific resource pool
3. Create/alter a user-defined function which returns the name of a workload group based on connection related information
4. Register the UDF with Resource Governor
5. Start the Resource Governor



Non-Uniform Memory Access (NUMA)

- Current trend in processor hardware design
- Increases scalability by reducing front-side bus contention
- CPUs and memory are divided into NUMA nodes
- A foreign memory access is slower than a local one (NUMA ratio) (Coreinfo v3.31)



Leveraging Resource Governor in SQL Server

Resource Governance

What Resources Can be Governed at What Level?

Resource	Resource Pool	Workload Group
CPU	MIN/MAX (%) CAP_CPU_Percent AFFINITY (SCHEDULER) AFFINITY (NUMANODE)	REQUEST_MAX_CPU_TIME_SEC MAX_DOP
Memory	MIN / MAX	REQUEST_MAX_MEMORY_GRANT_PERCENT REQUEST_MEMORY_GRANT_TIMEOUT_SEC
IO	MIN_IOPS_PER_VOLUME MAX_IOPS_PER_VOLUME	
Priority		IMPORTANCE { LOW | MEDIUM | HIGH }
Concurrency		GROUP_MAX_REQUESTS

SQL Server 2012 Feature

SQL Server 2014 Feature

CPU Consumption Summary

- MAX can be exceeded
- Calculation of CPU is per scheduler, think of it as divided up quantums of the CPU
- Design is based on scheduler activity, pool settings, and only if there is no CPU contention
- Importance not the same as priority

Note: With SQL Server 2012 you can provide more complete isolation of CPU resources for workloads and put caps on CPU usage for a higher level of predictability, and you can control a greater proportion of SQL Server memory allocations.

CPU Consumption Example

Pools	Min	Max	Max Total Shared	Effective Max	Default Pool
Pool A	30	90	40	70	40
Pool B	25	50	40	50	40
Pool C	5	80	40	45	40

- Total Shared % = 100 - SUM (all Min%)
  - **Example:** Total Shared% = 100 - (30%+25%+5%) = 100 - 60% = 40%
- Effective Max% for a pool = MIN(Pool Max%, Pool Min% + Total Shared%)
  - **Example:** Pool C, Eff. Max% = MIN (80%, 5%+40%) = 45%
- Be careful when setting MIN values since those are always guaranteed and can affect performance for other Pools

Memory Consumption Summary

- Resource Pools
  - MIN_MEMORY_PERCENT
  - MAX_MEMORY_PERCENT
- Workload Groups
  - REQUEST_MAX_MEMORY_GRANT_PERCENT
  - REQUEST_MEMORY_GRANT_TIMEOUT_SEC
  - MAX_MEMORY_GRANT_PERCENT

The Resource Governor can control more of SQL Server's memory as a result of the memory redesign in SQL Server 2012

IO Consumption Summary

- The ability to govern IO is a critical component for SQL Server performance
- Completes the resource needs for SQL Server
- Allows full isolation of resources between workloads critical for:
  - Hosting Companies
  - Workload consolidation by IT
- Categorization is critical for modern workloads including large scale deployments, helping to address:
  - Rogue workloads
  - Throw away queries
  - Maintenance operations

MAX_OUTSTANDING_IO_PER_VOLUME

```
ALTER RESOURCE GOVERNOR
{ DISABLE | RECONFIGURE } |
WITH ( CLASSIFIER_FUNCTION = { schema_name.function_name |
NULL } ) |
RESET STATISTICS |
WITH ( MAX_OUTSTANDING_IO_PER_VOLUME = value )
[ ; ]
```

'MAX_OUTSTANDING_IO_PER_VOLUME' is a knob where we ensure that the minimum IOPs do not compromise the disk.

There is no recommended value (max=100).
It is recommended to test performance with different values (Example: SQLIO)

Governing IO with Resource Governor

```
-- Set the classifier function and enable RG.
ALTER RESOURCE GOVERNOR
WITH (CLASSIFIER_FUNCTION = dbo.fnUserClassifier);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO

-- Set default values for the resource pools
-- so that IO RG is enabled.
ALTER RESOURCE POOL Customer1Pool WITH (MIN_IOPS_PER_VOLUME=0,
MAX_IOPS_PER_VOLUME=2147483647);
ALTER RESOURCE POOL Customer2Pool WITH (MIN_IOPS_PER_VOLUME=0,
MAX_IOPS_PER_VOLUME=2147483647);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

Leveraging Resource Governor in SQL Server

Monitoring Techniques

Resource Governor Monitoring Techniques

- Dynamic Management Views
  - sys.dm_resource_governor_resource_pools
  - sys.dm_resource_governor_resource_pool_volumes
- Extended Events
  - File_read_enqueued
  - File_write_enqueued
- Performance Monitor
- Resource Governor allows monitoring of SQL Server usage at both the Resource Pool and Workload Group levels
- Monitor before limiting!

Monitoring Resource Governor - Perfmon

- RG Instance per Pool: SQL Server: Resource Pool Stats
- RG Instance per Group: SQL Server: Workload Group Stats
- 6 new Performance Counters for IO
  - Track read/write IOs and bytes
  - Track read/write latency (average ms.)

New Resource Governor Performance Counters

Counter	Definition
Disk Read IOs/sec	Number of read operations from the disk in the last second
Disk Read Bytes/Sec	Number of bytes read from the disk in the last second
Avg. Disk ms/Read IO	Average time, in Milliseconds, of a read operation from disk
Disk Write IOs/sec	Number of write operations to the disk in the last second
Disk Write Bytes/sec	Number of bytes written to the disk in the last second
Avg. Disk ms/write IO	Average time, in Milliseconds, of a write operation from disk

Monitoring Resource Governor – Extended Events

- Two new **XEvents** to trace requests to the IO Resource Manager queues (**file_write_enqueued** and **file_read_enqueued**)
- Use these together with the events that trace issuing IO (file_read/file_written) and IO completion (file_read/write_completed)

Event Name	Data
file_read_enqueued	Mode File_handle Offset Database_id File_id Filegroup_id Size Path

Event Name	Data
file_write_enqueued	Mode File_handle Offset Database_id File_id Filegroup_id Size Path Io_data

Note: These two events are to trace enqueueing IO to the Resource Manager. Using these XE with the events that trace IO completion, users can investigate the IO subsystem

Monitoring Resource Governor - DMVs

- DMVs
  - sys.dm_resource_governor_workload_groups
  - sys.dm_resource_governor_resource_pools
    - New columns for I/O max, I/O read/write queue, stalls, and violations
  - sys.dm_resource_governor_resource_pool_volumes (2014+)
    - Per pool: read/write queued, issued, completed, and violations
  - sys.dm_resource_governor_configuration
    - New columns for I/O stall read/write latency
- Catalog Views
  - sys.resource_governor_configuration
  - sys.resource_governor_resource_pools
  - sys.resource_governor_workload_groups

Resource Governor Best Practices

- Enable the remote Dedicated Admin Connection (DAC)
- Optimize the Classifier Function
  - Use the PreConnect() Trace Events
  - Test the Classifier Function Under Load
  - Avoid Lookup Tables
  - Avoid Expensive String Operations
- Place Reasonable Limits on the Default Pool and Group
- Use Memory Constraints with Caution
- Set up Resource Governor with no limits first and then monitor with performance monitor before putting any caps in place

References

- Using the Resource Governor (White Paper)
  http://download.microsoft.com/download/D/B/D/DBDE7972-1EB9-470A-BA18-58849DB3EB3B/ResourceGov.docx
- Troubleshooting Resource Governor
  http://technet.microsoft.com/en-us/library/cc627395(v=SQL.105).aspx
- Considerations for Writing a Classifier Function
  http://technet.microsoft.com/en-us/library/bb933865(v=SQL.105).aspx
- IO Governance in SQL Server 2014
  http://blogs.technet.com/b/dataplatforminsider/archive/2013/11/07/io-resource-governance-in-sql-server-2014.aspx
- Resource Governor Whitepapers
  http://technet.microsoft.com/en-us/library/cc627395(v=SQL.105).aspx
- Configure Resource Governor Using a Template
  http://msdn.microsoft.com/en-us/library/bb934150.aspx
- SQL Server 2014 Developer Training Kit
  http://www.microsoft.com/en-us/download/details.aspx?id=41704