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

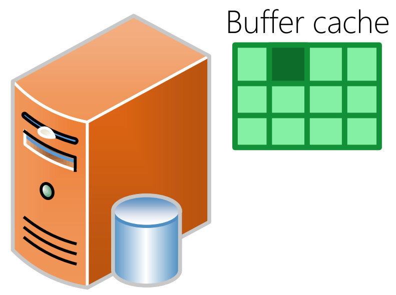
SQL Server Disk I/O Overview

SQL Server Disk I/O

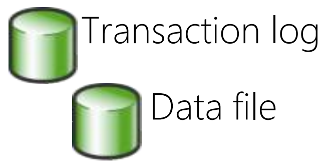
- Database page reads and writes always go through the buffer pool
- With on-disk table operations, SQL Server uses Write-Ahead logging in order to adhere to ACID principles

1. Data modification statement sent by application

2. Modification recorded in the log cache



3. Affected data pages are located in or read into the buffer cache and modified
4. Log cache record is flushed to the transaction log



5. Dirty pages are written to disk

SQL Server Disk I/O

Disk Patterns

SQL Server Disk I/O Patterns: Transaction Log

- One .ldf file per database
- Sequential reads and writes
- Write activity during log buffer flush operations
- Read activity during checkpoints, backups and recovery
- Features such as database mirroring and replication will increase read and write activity

- *You can have more than one log file per database, but it is not recommended. Since log files are written to sequentially, multiple log files will not provide any performance benefit

SQL Server Disk I/O Patterns: Data File(s)

- One .mdf file per database
- May have one or more .ndf files
- Random reads and writes
- Write activity during checkpoints, recovery and lazy writes
- Read activity during backups, other activity varies depending on query activity and buffer pool size

Checkpoints

- Flushes dirty pages (in-memory modified) from buffer pool to disk. Frequency of checkpoints varies based on database activity and recovery interval
- Four types of checkpoints:
  - **Automatic (default)** – database engine issues checkpoints automatically based on the server level "recovery interval" configuration option
  - **Indirect (new in SQL 2012)** – database engine issues checkpoints automatically based on the database level TARGET_RECOVERY_TIME
  - **Manual** – issued in the current database for your connection when you execute the T-SQL CHECKPOINT command
  - **Internal** – issued by various server operations

Indirect Checkpoints

- Changed algorithm:
- Based on number of dirty pages, not number of transactions
- More predictable database recovery time
- Does not directly write out dirty pages
- Leverages a background Recovery Writer thread:
- Serves to smooth out I/O spikes that would occur during normal checkpoints
- May increase average I/O

SQL Server Disk I/O

Identifying I/O Issues

Notable Disk Performance Counters

- Logical Disk: Avg. Disk Sec/Read and Logical Disk: Avg. Disk Sec/Write
  - Less than 10 ms = Very Good
  - Between 10-20 ms = Fair
  - Between 20-50 ms = Poor, needs attention
  - Greater than 50 ms = Serious I/O bottleneck
- Logical Disk: Avg. Disk Bytes/Transfer for knowledge on I/O sizes
- Logical Disk: Disk Transfers/sec for knowledge on IOPs

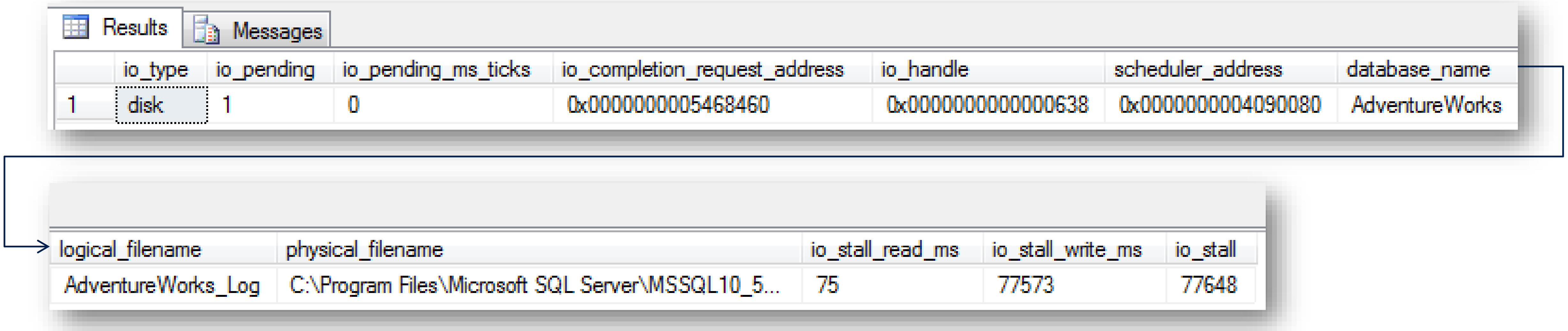
Notable Waits (sys.dm_os_wait_stats)

- ASYNC_IO_COMPLETION - Occurs when a task is waiting for I/Os to finish.
- IO_COMPLETION - Generally represents non-data page I/Os.
- LOGMGR - Occurs when a task is waiting for any outstanding log I/Os to finish before shutting down the log while closing the database.
- WRITELOG - Occurs while waiting for a log flush to complete. Common operations that cause log flushes are checkpoints and transaction commits.
- PAGEIOLATCH_x - Occurs when a task is waiting on a latch for a buffer that is in an I/O request. Only data pages.

Pending I/O DMV

sys.dm_io_pending_io_requests

- Returns a row for each in-flight pending I/O request:
  - Disk
  - Network



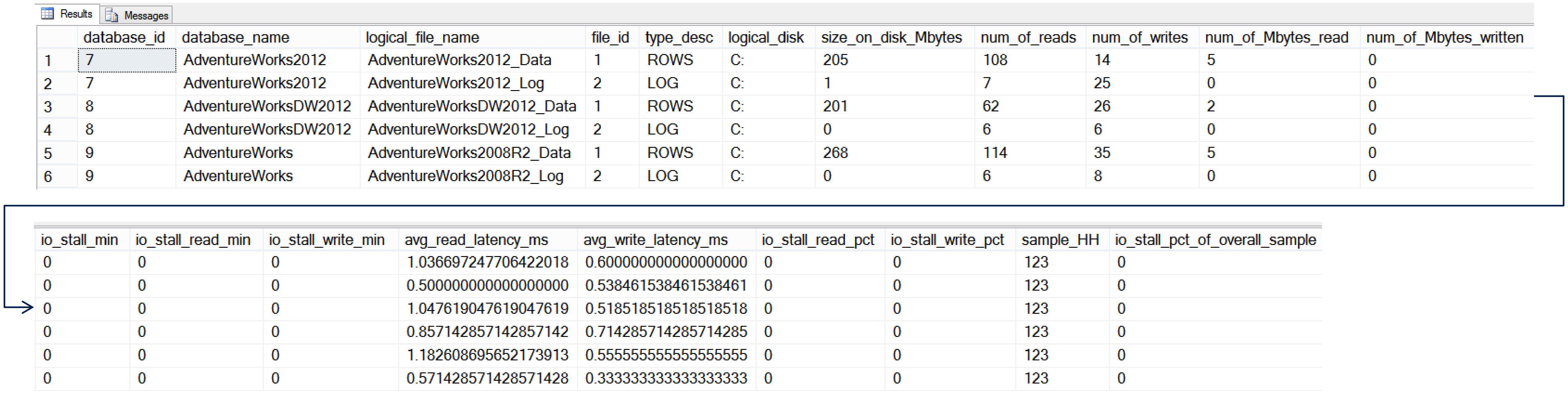
	io_type	io_pending	io_pending_ms_ticks	io_completion_request_address	io_handle	scheduler_address	database_name
1	disk	1	0	0x0000000005468460	0x0000000000000638	0x0000000004090080	AdventureWorks

logical_filename	physical_filename	io_stall_read_ms	io_stall_write_ms	io_stall
AdventureWorks_Log	C:\Program Files\Microsoft SQL Server\MSSQL10_5...	75	77573	77648

I/O Per File DMV

sys.dm_io_virtual_file_stats

- Returns I/O statistics for data and log files since the respective file was first attached to SQL Server



	database_id	database_name	logical_file_name	file_id	type_desc	logical_disk	size_on_disk_Mbytes	num_of_reads	num_of_writes	num_of_Mbytes_read	num_of_Mbytes_written	io_stall_min	io_stall_read_min	io_stall_write_min	avg_read_latency_ms	avg_write_latency_ms	io_stall_read_pct	io_stall_write_pct	sample_HH	io_stall_pct_of_overall_sample
1	7	AdventureWorks2012	AdventureWorks2012_Data	1	ROWS	C:	205	108	14	5	0	0	0	0	1.036697247706422018	0.600000000000000000	0	0	123	0
2	7	AdventureWorks2012	AdventureWorks2012_Log	2	LOG	C:	1	7	25	0	0	0	0	0	0.500000000000000000	0.538461538461538461	0	0	123	0
3	8	AdventureWorksDW2012	AdventureWorksDW2012_Data	1	ROWS	C:	201	62	26	2	0	0	0	0	1.047619047619047619	0.518518518518518518	0	0	123	0
4	8	AdventureWorksDW2012	AdventureWorksDW2012_Log	2	LOG	C:	0	6	6	0	0	0	0	0	0.857142857142857142	0.714285714285714285	0	0	123	0
5	9	AdventureWorks	AdventureWorks2008R2_Data	1	ROWS	C:	268	114	35	5	0	0	0	0	1.182608695652173913	0.555555555555555555	0	0	123	0
6	9	AdventureWorks	AdventureWorks2008R2_Log	2	LOG	C:	0	6	8	0	0	0	0	0	0.571428571428571428	0.333333333333333333	0	0	123	0

Example: sys.dm_io_virtual_file_stats

```
SELECT a.io_stall, a.io_stall_read_ms, a.io_stall_write_ms,
a.num_of_reads, a.num_of_writes,
( ( a.size_on_disk_bytes / 1024 ) / 1024.0 ) AS size_on_disk_mb,
db_name(a.database_id) AS dbname, b.name, a.file_id,
db_file_type = CASE
 WHEN a.file_id = 2 THEN 'Log'
 ELSE 'Data'
 END,
UPPER(SUBSTRING(b.physical_name, 1, 2)) AS disk_location
FROM sys.dm_io_virtual_file_stats (NULL, NULL) a
JOIN sys.master_files b ON a.file_id = b.file_id
AND a.database_id = b.database_id
ORDER BY a.io_stall DESC
```

SQLIO

SQLIO is a tool to determine the I/O capacity of a given configuration.

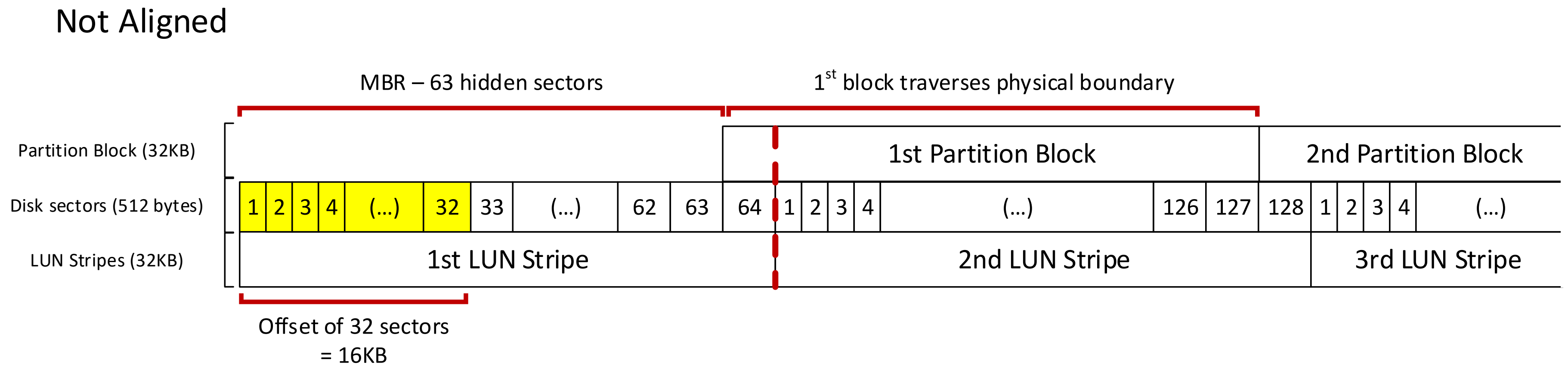
- To test a particular volume limits, we need to:

1. Create a test file of appropriate size, which should be large files
2. Run the tool with different settings to test a wide range of I/O profiles
3. Tune queue depth for optimal latency vs. IOPs with the results

I/O Troubleshooting Considerations

Scenario	Symptom	Diagnosis	Solution
Slow overall performance	Disk performance is bad at all times	Read error log for long I/O messages	Identify I/O inefficient queries Review disk system configurations (HBA Queues, etc.)
Slow overall performance with a foreseeable patterns	Disk performance gets worse at specific intervals	Identify disk latencies using Logical Disk: Avg. Disk Sec/Read and Write counters	If legitimate workload, re-architect to handle more IOPS
Slow performance on some workloads	Some database's workloads have performance issues, but not all	Identify via DMVs which files have pending I/Os	Stripe relevant databases

Partition Alignment



Lesson Knowledge Check

- When will an update to a row be written to the data file?
- What happens to a data modification if the SQL Server crashes before the change has been written to the data file?
- How can we know the I/O characteristics of data and log files?