Statistics

Statistics

- No auto update statistics (NORECOMPUTE)
- Auto create statistics are supported
- No filtered statistics
- Only FULLSCAN
- For large tables this can be problematic

Interop

- Manual update stats for good cardinality estimates
- sp_updatestats

Natively compiled procedures

- No recompiles or stats updates change the plan
- Drop and recreate the procedure
- Procedure uses UNKNOWN at compile time

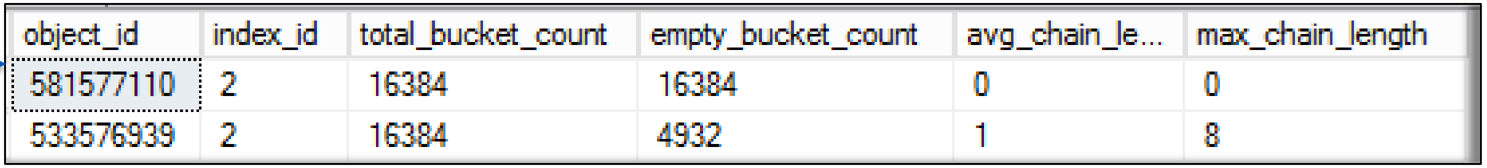
Indexing Guidelines – Hash Indexes

All memory-optimized table hash indexes are covering

Hash index:

- Bucket-count too low or too high impacts read and write behavior (decided at create)
- Bucket count sizing dependent on number of distinct rows for the indexed column

```
SELECT *
FROM sys.dm_db_xtp_hash_index_stats
```



object_id	index_id	total_bucket_count	empty_bucket_count	avg_chain_le...	max_chain_length
581577110	2	16384	16384	0	0
533576939	2	16384	4932	1	8

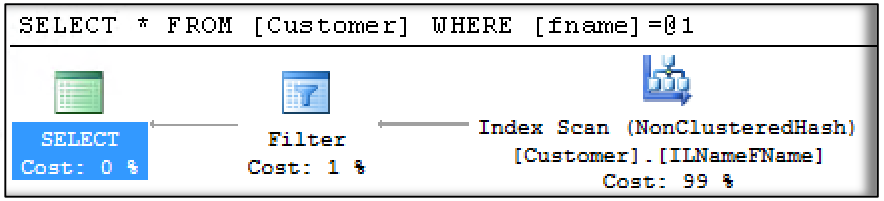
- Not qualifying the resultset (select with no predicate) will result in scanning all buckets

Indexing Guidelines – Hash Indexes

- Querying on sub-set of the columns in the hash index will result in scans

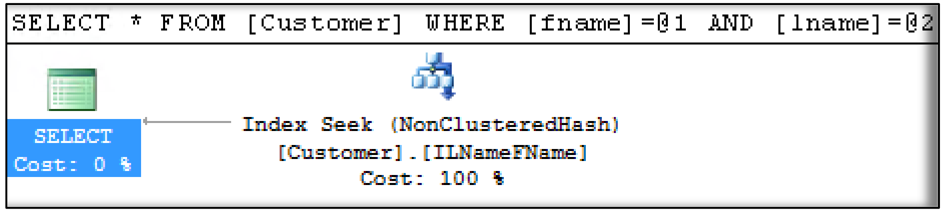
```
CREATE TABLE [Customer] ([CustomerID] INT NOT NULL,
         [AddressID] INT NOT NULL,
         [fName] NVARCHAR(250) COLLATE Latin1_General_100_BIN2 NOT NULL,
         [LName] NVARCHAR(250) COLLATE Latin1_General_100_BIN2 NOT NULL,
INDEX [ILNameFName] NONCLUSTERED HASH (LName, fName) WITH (BUCKET_COUNT = 16384),
PRIMARY KEY NONCLUSTERED HASH (CustomerID) WITH (BUCKET_COUNT = 131072))
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
```

```
SELECT * FROM Customer
WHERE fname = ‘Dirk’
```



```
SELECT * FROM [Customer] WHERE [fname]=@1
```

```
SELECT * FROM Customer
WHERE fname = ‘Dirk’
AND lname = ‘Nowitzky’
```



```
SELECT * FROM [Customer] WHERE [fname]=@1 AND [lname]=@2
```

- Many duplicates in the index can cause hash collisions which is expensive, use range indexes
- Does not support ordered scans

Indexing Guidelines – Nonclustered Indexes

Ideal for Range Scans

Nonclustered (ordered):

- No bucket-count needed
- Only does a “forward” scan – Don’t want to ORDER BY c1 DESC on column defined as: c1 ASC in index definition
- May have to create both descending and ascending indexes if needed
- Support search on leading columns
- Choose if doing index seek on inequality predicate

Index Guidelines Summary

Point Lookups

- Full index key
- Hash index

Range Lookups

- Sorted datasets
- Non-clustered

Both Range and Point Lookups

- Create two indexes
- Hash and non-clustered

String Comparison and Collation

Indexing on character columns requires BIN2 collation

```
Msg 12328, Level 16, State 107, Line 21
Indexes on character columns that do not use a *_BIN2 collation are not supported with indexes on memory optimized tables.
Msg 1750, Level 16, State 0, Line 21
Could not create constraint or index. See previous errors.
```

Natively compiled procedures can compare character columns only to BIN2 collation

```
Msg 12327, Level 16, State 106, Procedure proc_test, Line 17
Comparison, sorting, and manipulation of character strings that do not use a *_BIN2 collation is not supported with natively compiled stored procedures
```

Other Query Performance Considerations

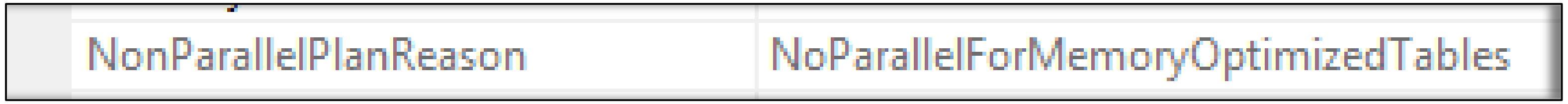
Interop

- All Join types supported
- Index Scan shown as Table Scan
- No Predicate pushdown, additional Filter will appear

Natively compiled

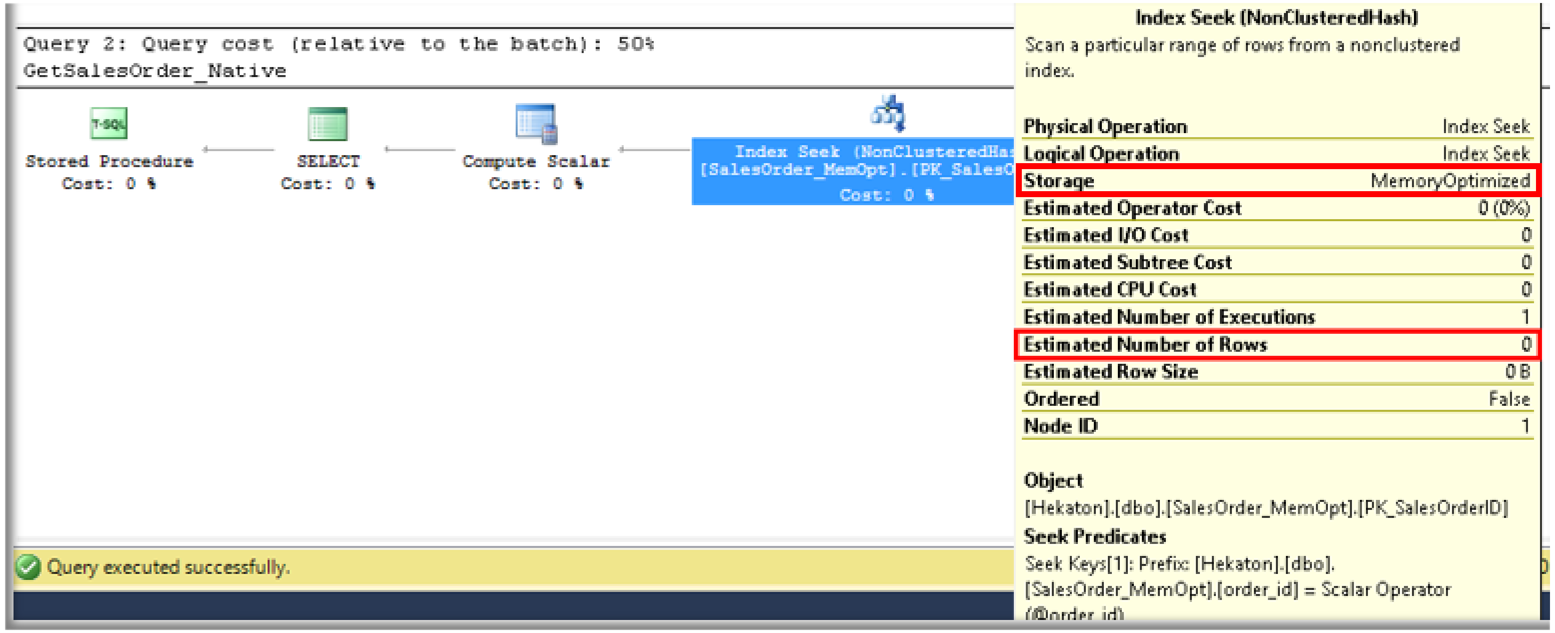
- Only Nested Loop join is used
- No join hints can be specified
- Only Stream Aggregation (No Hash Aggregation)
- No parameter sniffing

Serial Plan (MAXDOP = 1) for queries referencing memory-optimized tables



NonParallelPlanReason	NoParallelForMemoryOptimizedTables

Query Plan Changes



```
Query 2: Query cost (relative to the batch): 50%
GetSalesOrder_Native
```

Index Seek (NonClusteredHash)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Storage	MemoryOptimized
Estimated Operator Cost	0 (0%)
Estimated I/O Cost	0
Estimated Subtree Cost	0
Estimated CPU Cost	0
Estimated Number of Executions	1
Estimated Number of Rows	0
Estimated Row Size	0 B
Ordered	False
Node ID	1

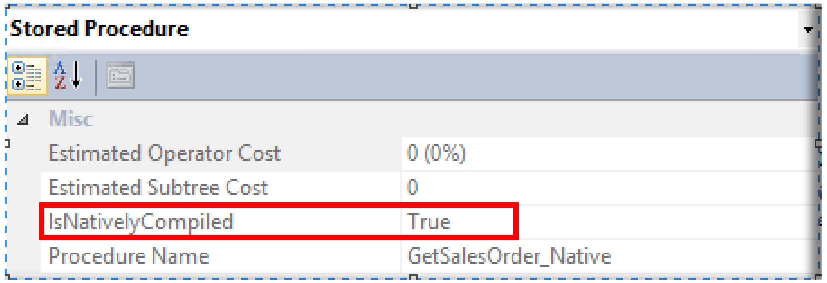
Object

[Hekaton].[dbo].[SalesOrder_MemOpt].[PK_SalesOrderID]

Seek Predicates

Seek Keys[1]: Prefix: [Hekaton].[dbo].
[SalesOrder_MemOpt].[order_id] = Scalar Operator
(@order_id)

Query executed successfully.



Stored Procedure

Misc

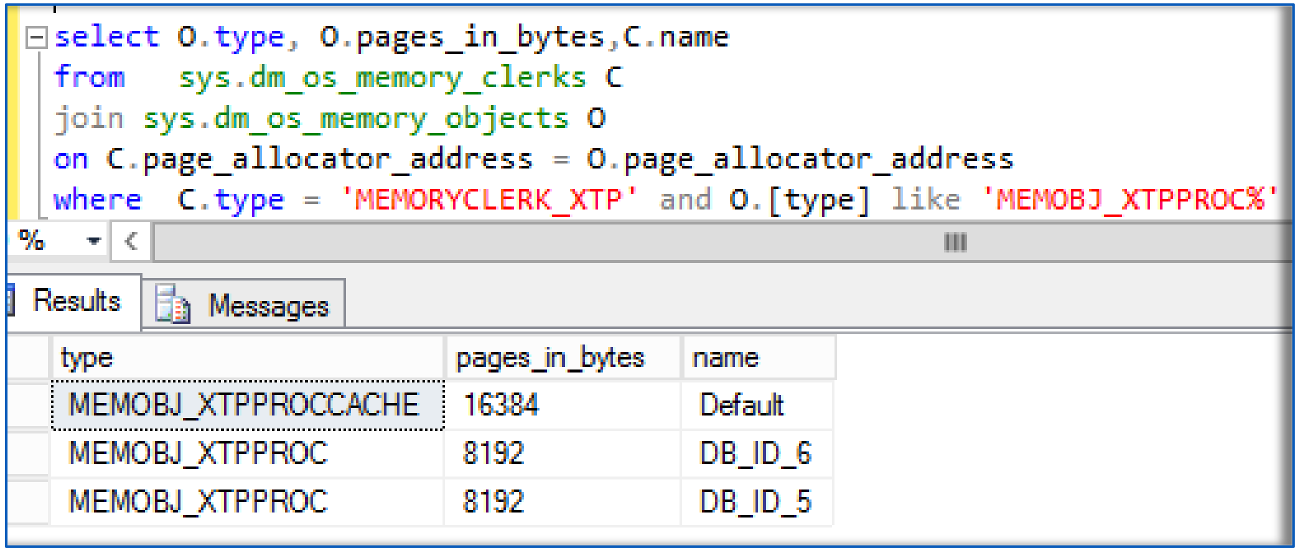
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0
IsNativelyCompiled	True
Procedure Name	GetSalesOrder_Native

- Natively compiled procedures can only display estimated showplan
- Estimates are always 0
- Actual plan can be obtained for interop
- Plan guides not supported for Natively compiled procedures

Plan Cache

Natively compiled procedures

- Cannot be returned by sys.dm_exec_query_plan DMF
- CACHESTORE_OBJCP/CACHESTORE_SQLCP does not account for them
- DBCC FREEPROCCACHE/ Free System Cache not applicable
- Requires drop and recreate as compiled plan is a DLL



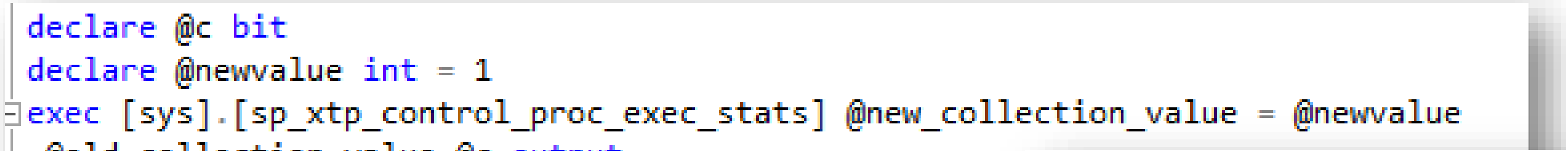
```
select O.type, O.pages_in_bytes,C.name
from   sys.dm_os_memory_clerks C
join sys.dm_os_memory_objects O
on C.page_allocator_address = O.page_allocator_address
where  C.type = 'MEMORYCLERK_XTP' and O.[type] like 'MEMOBJ_XTPPROC%'
```

type	pages_in_bytes	name
MEMOBJ_XTPPROCCACHE	16384	Default
MEMOBJ_XTPPROC	8192	DB_ID_6
MEMOBJ_XTPPROC	8192	DB_ID_5

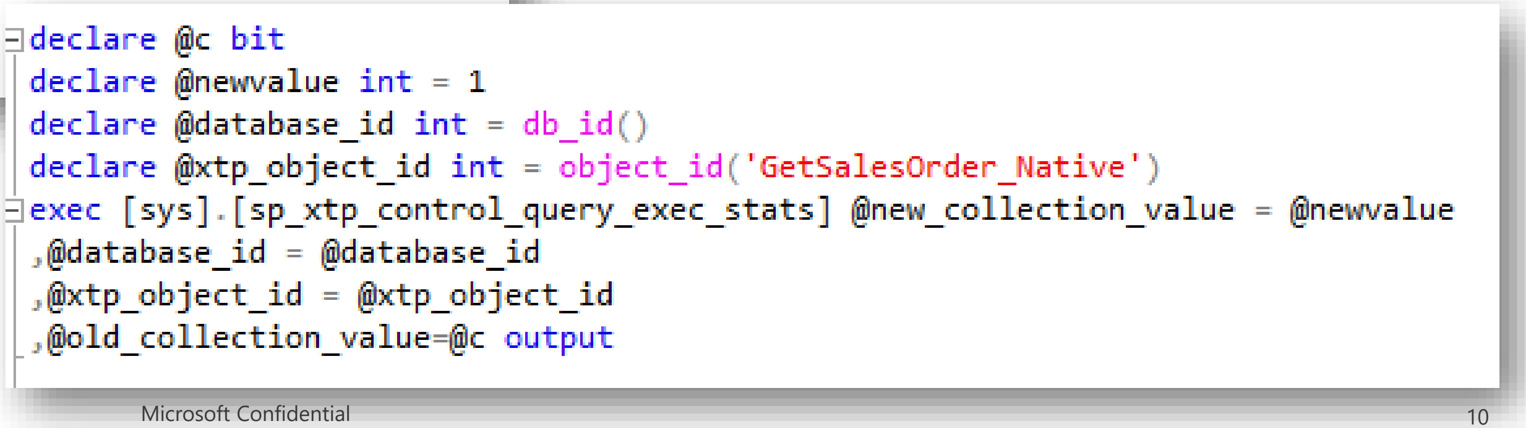
Execution Statistics Collection

Natively compiled procedures

- By default no collection in sys.dm_exec_query_stats / sys.dm_exec_procedure_stats
- Reset to off on SQL Server restart
- Performance impact if you enable stats collection
- Procedure stats enabled server wide with *sp_xtp_control_proc_exec_stats*
- Query Stats enabled per procedure with *sp_xtp_control_query_exec_stats*
- Only sp_statement_completed (no starting) event produced in Xevent



```
declare @c bit
declare @newvalue int = 1
exec [sys].[sp_xtp_control_proc_exec_stats] @new_collection_value = @newvalue
,@old_collection_value=@c output
select @newvalue as NewValue, @c as 'OldValue'
```



```
declare @c bit
declare @newvalue int = 1
declare @database_id int = db_id()
declare @xtp_object_id int = object_id('GetSalesOrder_Native')
exec [sys].[sp_xtp_control_query_exec_stats] @new_collection_value = @newvalue
,@database_id = @database_id
,@xtp_object_id = @xtp_object_id
,@old_collection_value=@c output
```

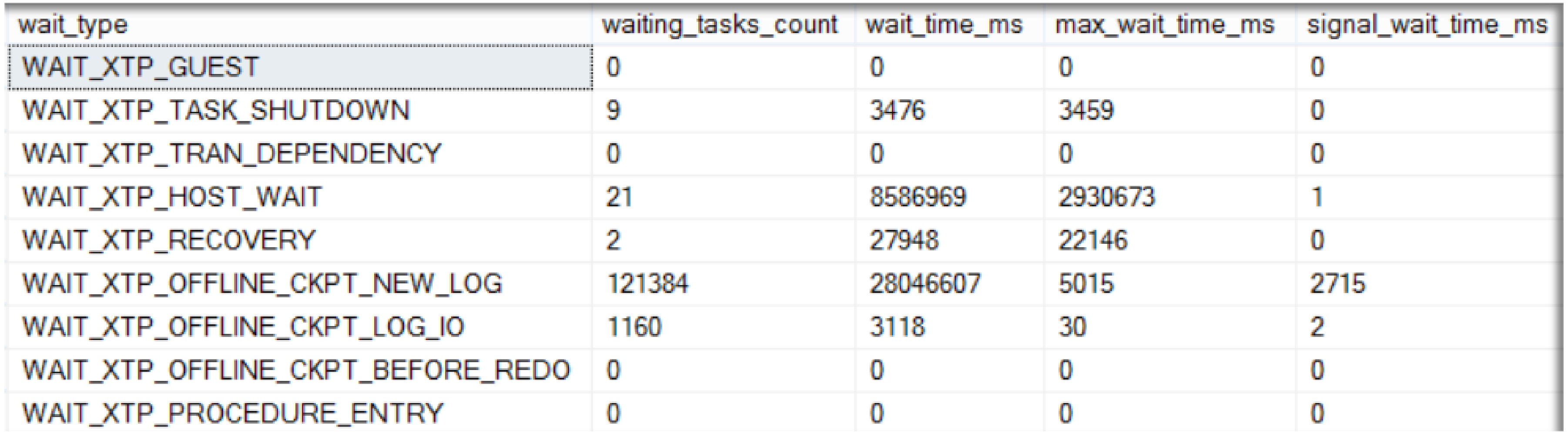
XTP (In-Memory OLTP) Performance Counters

Performance object	Description
XTP Cursors	The XTP Cursors performance object contains counters related **to internal XTP engine cursors**. Cursors are the low-level building blocks the XTP engine uses to process Transact-SQL queries. As such, you do not typically have direct control over them.
XTP Garbage Collection	The XTP Garbage Collection performance object contains counters related to the XTP engine's **garbage collector**.
XTP Phantom Processor	The XTP Phantom Processor performance object contains counters related to the XTP engine's phantom processing subsystem. This component is responsible for **detecting phantom rows in transactions running at the SERIALIZABLE isolation level**.
XTP Transaction Log	The XTP Transaction Log performance object contains counters related to **XTP transaction logging** in SQL Server.
XTP Transactions	The XTP Transactions performance object contains counters related to **XTP engine transactions** in SQL Server.

sys.dm_os_wait_stats

Global Wait stats

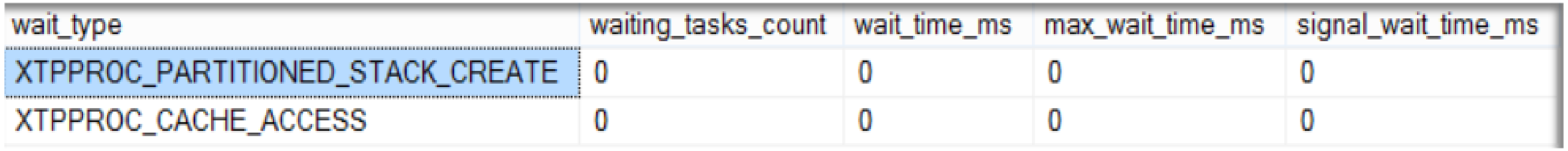
- WAIT_XTP_*



wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
WAIT_XTP_GUEST	0	0	0	0
WAIT_XTP_TASK_SHUTDOWN	9	3476	3459	0
WAIT_XTP_TRAN_DEPENDENCY	0	0	0	0
WAIT_XTP_HOST_WAIT	21	8586969	2930673	1
WAIT_XTP_RECOVERY	2	27948	22146	0
WAIT_XTP_OFFLINE_CKPT_NEW_LOG	121384	28046607	5015	2715
WAIT_XTP_OFFLINE_CKPT_LOG_IO	1160	3118	30	2
WAIT_XTP_OFFLINE_CKPT_BEFORE_REDO	0	0	0	0
WAIT_XTP_PROCEDURE_ENTRY	0	0	0	0

Native Procedure

- XTPPROC_*



wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
XTPPROC_PARTITIONED_STACK_CREATE	0	0	0	0
XTPPROC_CACHE_ACCESS	0	0	0	0

Storage Considerations

	Capacity	Performance
Data	Capacity needed is 2-3 x size of durable memory optimized tables Around 250GB max per database On-disk size maybe larger than size in memory	Sequential IO bandwidth sufficient to meet RTO 64KB NFTS Cluster Spread into multiple containers (Odd number) SSD or Fast Spinning media (15K SAS array)
Log	Per transaction log consumption is less than disk based tables	Latency is VERY important for durable tables SSDs (PCIe) Consider Delayed Durability or Non Durable consideration

Memory Considerations

Sizing	Configure Max Server memory appropriately	Size of in-memory table + indexes + row versions Consider other workloads on the instance Short transactions (versions)
Managing	Resource Governor Define dedicated pool	Short transactions (versions) Force Garbage Collection

Other Considerations

CPU

Number of Cores/Sockets
2-4 Socket boxes
< 60 cores

Resource Governor can help here as well

Configuration

Instant File Initialization
Anti-Virus Exclusions