Standard Operating Procedure (SOP) for configuring and managing **MAXDOP (Maximum Degree of Parallelism)** on Microsoft SQL Server, aligned with the content of the provided article and incorporating *industry best practices* for SQL Server 2017 through SQL Server 2025.

This SOP is suitable for database administrators and engineers responsible for performance tuning in production environments.

Standard Operating Procedure (SOP)

Title: SQL Server MAXDOP Configuration and Best Practices
Scope: Applies to Microsoft SQL Server instances (on-premises and cloud) from 2017 through 2025
Audience: DBAs, System Architects, Performance Engineers
Objective: To define the process for evaluating, configuring, testing, and maintaining the MAXDOP setting to optimize query parallelism while minimizing CPU contention and performance degradation.

1. Purpose
MAXDOP determines the maximum number of CPU cores SQL Server can use to execute a single query plan in parallel. Proper configuration of MAXDOP:

- Improves query performance where appropriate.
- Reduces excessive CPU usage and resource contention.
- Ensures workload stability across OLTP and OLAP systems.

2. Definitions
MAXDOP: Maximum Degree of Parallelism; a SQL Server configuration setting.
NUMA: Non-Uniform Memory Access; affects memory and CPU grouping.
CXPACKET waits: Wait type associated with parallel threads waiting on each other.
Cost Threshold for Parallelism: Value that defines when SQL Server will consider parallel execution. Microsoft Learn
https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option?view=sql-server-ver17

3. Prerequisites
Prior to implementing changes:

- Ensure you have *appropriate administrative privileges* on the SQL Server instance.
- Collect baseline performance metrics (CPU, wait stats, query durations).
- Understand the workload type (OLTP vs. OLAP).

4. Guiding Principles
4.1 Start with Baseline and Monitoring
Before altering MAXDOP:

1. Monitor current performance counters (CPU utilization, CXPACKET waits).
2. Identify heavy queries using tools such as *Query Store*, *sp_BlitzCache*, *Extended Events*, or similar.

4.2 Understand Default Behavior

- Default MAXDOP = 0 (unlimited) allows SQL Server to use all cores — *not recommended for most production systems*.

https://techcommunity.microsoft.com/blog/azuresqlblog/changing-default-maxdop-in-azure-sql-database-and-azure-sql-managed-instance/1538528

5. MAXDOP Configuration Guidelines
Apply the following **Microsoft and industry best practices** when setting MAXDOP:
https://docs.aws.amazon.com/prescriptive-guidance/latest/sql-server-ec2-best-practices/maxdop.html

5.1 Single NUMA Node Systems

Logical Processors	Recommended MAXDOP
≤ 8	≤ number of logical processors
> 8	8

5.2 Multiple NUMA Node Systems

Logical Processors per NUMA Node	Recommended MAXDOP
≤ 16	≤ number of logical processors per NUMA node
> 16	Half of logical processors per NUMA node — *max 16*

6. Adjustments by Workload Type (Optional Tuning)

After initial configuration and performance testing:

- **OLTP Workloads:**
  - Lower MAXDOP values (e.g., 1–4) to favor concurrency and reduce contention.
  - Raise *Cost Threshold for Parallelism* to 25-50 to prevent trivial parallel plans. LinkedIn

https://www.linkedin.com/posts/utkarsh-kumar-718a0638_as-a-sql-server-dba-two-of-the-most-overlooked-activity-7353458501983113216--txh/

- **OLAP / Data Warehousing Workloads:**
  - Allow higher MAXDOP to maximize throughput for analytical queries.
  - Raise *Cost Threshold for Parallelism* higher (e.g., 100-500).

7. Override Options

You can override the instance MAXDOP setting at finer granularity:

- **Query level:** OPTION (MAXDOP n)
- **Database level:** Database-scoped configuration
- **Workload level:** Resource Governor workload group setting

Use overrides only when necessary to address specific performance cases.

8. Implementation Steps

8.1 Review Current Settings

```
SELECT name, value, value_in_use
FROM sys.configurations
WHERE name IN ('max degree of parallelism', 'cost threshold for parallelism');
```

8.2 Change MAXDOP

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'max degree of parallelism', <value>;
RECONFIGURE;
```

8.3 Set Cost Threshold for Parallelism

```
EXEC sp_configure 'cost threshold for parallelism', <value>;
RECONFIGURE;
```

8.4 Disable Advanced Options (Optional)

```
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;
```

9. Testing and Validation

After implementing changes:

1. Monitor CPU utilization.
2. Evaluate wait statistics (especially *CXPACKET* and *Threads*).
3. Review query execution times for regression or improvement.
4. Validate that SLA requirements continue to be met.

Do not deploy changes directly into production without prior testing in a staging environment.

10. Ongoing Monitoring

- Regularly review performance metrics and revise MAXDOP as workloads evolve.
- Periodically re-evaluate Cost Threshold and other parallelism settings.
- Ensure metrics are collected after significant changes (e.g., hardware upgrades, schema changes).

11. Documentation and Change Control

Each change to MAXDOP must be documented with:

- Reason for change
- Pre- and post-change performance metrics
- Rollback procedures
- Approval by DBA lead or architect

12. Safety and Risk Mitigation

- Always schedule production changes during maintenance windows.
- Maintain backup and rollback scripts.
- Ensure stakeholders are aware of potential performance impact.

13. Revision History

Version	Date	Description
1.0	2025-12-25	Initial SOP based on SQL Server best practices

This SOP aligns the *technical content from the referenced article* with Microsoft documentation and industry recommendations on MAXDOP configuration.

Reference:

https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option?view=sql-server-ver17

https://techcommunity.microsoft.com/blog/azuresqlblog/changing-default-maxdop-in-azure-sql-database-and-azure-sql-managed-instance/1538528

https://docs.aws.amazon.com/prescriptive-guidance/latest/sql-server-ec2-best-practices/maxdop.html

https://www.techearth.xyz/post/2025/05/06/maxdop-what-you-need-to-know

Operational artifacts to accompany the MAXDOP SOP for SQL Server 2017–2025.
These are designed for production change control, safe rollback, and post-change monitoring.

1. Change Request (CR) Template – MAXDOP Configuration

Change Title
MAXDOP and Cost Threshold for Parallelism Configuration Update

Change Type
☐ Standard ☐ Normal ☐ Emergency

Environment
☐ Production ☐ Pre-Production ☐ UAT ☐ Development

SQL Server Version
(SQL Server 2017 / 2019 / 2022 / 2025)

Instance Name
<ServerName>\<InstanceName>

Current Configuration

- MAXDOP: <current value>
- Cost Threshold for Parallelism: <current value>
- NUMA Nodes: <count>
- Logical CPUs per NUMA Node: <count>

Proposed Configuration

- MAXDOP: <new value>
- Cost Threshold for Parallelism: <new value>

Business Justification

- Reduce CPU contention / CXPACKET waits
- Align with Microsoft and industry best practices
- Improve workload stability and query performance

Risk Assessment

- Risk Level: ☐ Low ☐ Medium ☐ High
- Potential Impact: Temporary performance variation during plan recompilation

Testing Performed

- Baseline CPU, waits, and query duration collected
- Tested in non-production environment

Implementation Window

- Date:
- Start Time:
- End Time:

Rollback Plan

- Revert to previous MAXDOP and Cost Threshold values (script attached)

Approvals

- DBA Lead:
- Application Owner:
- Change Manager:

2. Pre-Change Baseline Collection Script

Run this **before** making any changes and save results.

```
-- Current configuration
SELECT name, value, value_in_use
FROM sys.configurations
WHERE name IN (
   'max degree of parallelism',
   'cost threshold for parallelism'
);

-- CPU and NUMA details
SELECT
   cpu_count,
   hyperthread_ratio,
   numa_node_count
FROM sys.dm_os_sys_info;

-- Top waits (focus on parallelism)
SELECT *
FROM sys.dm_os_wait_stats
WHERE wait_type LIKE 'CX%'
ORDER BY wait_time_ms DESC;

-- Top CPU-consuming queries (Query Store required)
SELECT TOP (10)
   qsqt.query_sql_text,
   SUM(qsrs.avg_cpu_time * qsrs.count_executions) / NULLIF(SUM(qsrs.count_executions), 0) AS avg_cpu_time,
   SUM(qsrs.avg_duration * qsrs.count_executions) / NULLIF(SUM(qsrs.count_executions), 0) AS avg_duration,
   SUM(qsrs.count_executions) AS execution_count
FROM sys.query_store_runtime_stats qsrs
JOIN sys.query_store_plan qsp
   ON qsrs.plan_id = qsp.plan_id
JOIN sys.query_store_query qsq
   ON qsp.query_id = qsq.query_id
JOIN sys.query_store_query_text qsqt
   ON qsq.query_text_id = qsqt.query_text_id
GROUP BY qsqt.query_sql_text
ORDER BY avg_cpu_time DESC;
```

3. Implementation Script (Production-Safe)

```
-- Enable advanced options
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
```

```
-- Set MAXDOP
EXEC sp_configure 'max degree of parallelism', <NEW_MAXDOP>;
RECONFIGURE;

-- Set Cost Threshold for Parallelism
EXEC sp_configure 'cost threshold for parallelism', <NEW_COST_THRESHOLD>;
RECONFIGURE;

-- Optional: lock advanced options again
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;
```

Operational Note

- No SQL Server restart is required.
- Expect plan cache churn; monitor immediately after change.

4. Rollback Script (Mandatory Attachment)

Use this script if performance degradation is observed.

```
-- Enable advanced options
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;

-- Restore previous values
EXEC sp_configure 'max degree of parallelism', <OLD_MAXDOP>;
RECONFIGURE;
```



```
EXEC sp_configure 'cost threshold for parallelism', <OLD_COST_THRESHOLD>;
RECONFIGURE;

-- Disable advanced options
EXEC sp_configure 'show advanced options', 0;
RECONFIGURE;
```

5. Post-Change Validation & Monitoring Checklist

Execute within **15 minutes**, **1 hour**, and **24 hours** after change.

5.1 Configuration Validation

```
SELECT name, value_in_use
FROM sys.configurations
WHERE name IN (
   'max degree of parallelism',
   'cost threshold for parallelism'
);
```

5.2 Parallelism and CPU Health

```
-- Parallelism waits
SELECT *
FROM sys.dm_os_wait_stats
WHERE wait_type IN ('CXPACKET', 'CXCONSUMER');
```

```
-- CPU utilization snapshot
SELECT
   DATEADD(ms,
        x.[timestamp] - si.ms_ticks,
        si.sqlserver_start_time) AS event_time,
   x.[timestamp] AS ring_buffer_timestamp,
   x.record.value('(./Record/SchedulerMonitorEvent/SystemHealth/SystemIdle)[1]', 'int') AS SystemIdle,
   x.record.value('(./Record/SchedulerMonitorEvent/SystemHealth/ProcessUtilization)[1]', 'int') AS SQLCPU
FROM (
   SELECT
      [timestamp],
      CONVERT(xml, record) AS record
   FROM sys.dm_os_ring_buffers
   WHERE ring_buffer_type = 'RING_BUFFER_SCHEDULER_MONITOR'
    AND record LIKE '%<SystemHealth>%'
) AS x
CROSS JOIN sys.dm_os_sys_info AS si
ORDER BY event_time DESC;
```

5.3 Query Performance Regression Check

```
SELECT TOP (10)
   qsqt.query_sql_text,
   SUM(qsrs.avg_duration * qsrs.count_executions)
      / NULLIF(SUM(qsrs.count_executions), 0) AS avg_duration,
   SUM(qsrs.avg_cpu_time * qsrs.count_executions)
      / NULLIF(SUM(qsrs.count_executions), 0) AS avg_cpu_time,
   SUM(qsrs.count_executions) AS execution_count
FROM sys.query_store_runtime_stats qsrs
JOIN sys.query_store_plan qsp
   ON qsrs.plan_id = qsp.plan_id
JOIN sys.query_store_query qsq
   ON qsp.query_id = qsq.query_id
JOIN sys.query_store_query_text qsqt
   ON qsq.query_text_id = qsqt.query_text_id
GROUP BY qsqt.query_sql_text
ORDER BY avg_duration DESC;
```


6. Operational Decision Matrix (Quick Reference)

Observation	Action
High CXPACKET, low CPU	Increase Cost Threshold
High CPU, many concurrent queries	Lower MAXDOP
Long-running analytical queries	Allow higher MAXDOP
OLTP latency increase	Reduce MAXDOP to 2–4

7. Audit & Documentation Requirements

- Attach baseline and post-change metrics to CR
- Update internal DBA runbook
- Record change in CMDB or configuration repository

Production-ready implementations for both requests, aligned with Microsoft guidance and SQL Server 2017–2025 behavior.

1. PowerShell Script – Auto-Calculate Recommended MAXDOP

This script:

- Connects to a SQL Server instance
- Detects logical CPUs and NUMA configuration
- Calculates **recommended MAXDOP** using Microsoft best practices
- Outputs both the recommendation and the T-SQL needed to apply it

Logic Implemented (Microsoft Best Practice)

- If **single NUMA node**
  → MAXDOP = min(Logical CPUs, 8)
- If **multiple NUMA nodes**
  → MAXDOP = min(Logical CPUs per NUMA node, 16)

PowerShell Script

```
<#
.SYNOPSIS
   Calculates a recommended MAXDOP value for a SQL Server instance
   based on CPU and NUMA configuration.
#>
[CmdletBinding()]
param (
   [string]$SqlInstance = "localhost",
   [string]$OutputFile  = "D:\Scripts\MaxDop_Result.txt"
)
Set-StrictMode -Version Latest
$ErrorActionPreference = "Stop"

Import-Module SqlServer

$query = @"
SELECT
   cpu_count,
   hyperthread_ratio,
   numa_node_count
FROM sys.dm_os_sys_info;
"@

try {
   $result = Invoke-Sqlcmd `
      -ServerInstance $SqlInstance `
      -Query $query `
      -TrustServerCertificate `
      -Encrypt Optional
}
catch {
   throw "Failed to connect to SQL Server instance '$SqlInstance'. $_"
}

if (-not $result) {
```

```
    throw "Query returned no results from sys.dm_os_sys_info."
}
$cpuCount  = [int]$result.cpu_count
$numaNodes = [int]$result.numa_node_count

if ($numaNodes -le 1) {
    $recommendedMaxDop = [Math]::Min($cpuCount, 8)
}
else {
    $cpusPerNuma = [Math]::Floor($cpuCount / $numaNodes)
    $recommendedMaxDop = [Math]::Min($cpusPerNuma, 16)
}
$output = @"
SQL Instance       : $SqlInstance
Logical CPUs       : $cpuCount
NUMA Nodes         : $numaNodes
Recommended MAXDOP  : $recommendedMaxDop

T-SQL to apply:
EXEC sp_configure 'max degree of parallelism', $recommendedMaxDop;
RECONFIGURE;
"@
$output | Out-File -FilePath $OutputFile -Encoding UTF8

Write-Host "Output written to $OutputFile"
```



```
[PSCustomObject]@{
    SqlInstance      = $SqlInstance
    LogicalCPUs      = $cpuCount
    NumaNodes        = $numaNodes
    RecommendedMAXDOP = $recommendedMaxDop
    OutputFile       = $OutputFile
}
```

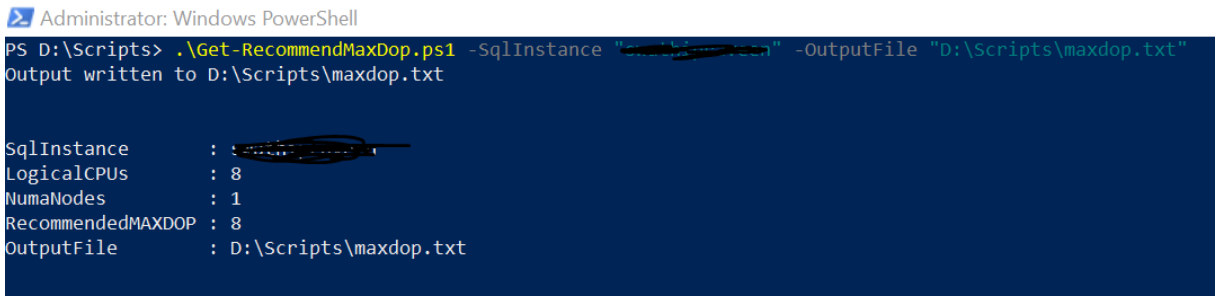
Example Output

SQL Instance: PROD-SQL01

Logical CPUs: 32

NUMA Nodes: 2

Recommended MAXDOP: 16



2. SQL Server Resource Governor – Workload-Specific MAXDOP Control

This example demonstrates:

- Limiting MAXDOP for **OLTP workloads**
- Allowing higher parallelism for **reporting / batch workloads**
- Classifying sessions using login or application name

Resource Governor is available in **Enterprise Edition** and **Azure SQL Managed Instance**.

2.1 Create Workload Groups

```
USE master;
GO
-- OLTP workload group (low parallelism)
CREATE WORKLOAD GROUP OLTP_Group
WITH
(
   MAX_DOP = 2,
   REQUEST_MAX_CPU_TIME_SEC = 0
);
GO

-- Reporting workload group (higher parallelism)
CREATE WORKLOAD GROUP Reporting_Group
WITH
(
   MAX_DOP = 8,
   REQUEST_MAX_CPU_TIME_SEC = 0
);
```


2.2 Create a Resource Pool (Optional but Recommended)

```
CREATE RESOURCE POOL UserPool
WITH
(
   MAX_CPU_PERCENT = 100,
   MIN_CPU_PERCENT = 0
);
GO

ALTER WORKLOAD GROUP OLTP_Group
USING UserPool;

ALTER WORKLOAD GROUP Reporting_Group
USING UserPool;
```

2.3 Create the Classifier Function

This example classifies:

- Application name containing **OLTP**
- Application name containing **REPORT**

```
CREATE FUNCTION dbo.ResourceGovernorClassifier()
RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
   DECLARE @group_name sysname;

   IF APP_NAME() LIKE '%OLTP%'
```

```
        SET @group_name = 'OLTP_Group';
    ELSE IF APP_NAME() LIKE '%REPORT%'
        SET @group_name = 'Reporting_Group';
    ELSE
        SET @group_name = 'default';

    RETURN @group_name;
END;
GO
```

2.4 Enable Resource Governor

```
ALTER RESOURCE GOVERNOR
WITH (CLASSIFIER_FUNCTION = dbo.ResourceGovernorClassifier);
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
```

2.5 Validation Queries

```
-- Verify workload groups
SELECT name, max_dop
FROM sys.resource_governor_workload_groups;

-- See active sessions and their workload group
SELECT
    s.session_id,
    s.login_name,
    s.program_name,
    g.name AS workload_group
FROM sys.dm_exec_sessions s
JOIN sys.resource_governor_workload_groups g
    ON s.group_id = g.group_id
WHERE s.is_user_process = 1;
```


3. Operational Best Practices

- Keep **instance-level MAXDOP conservative**
- Use Resource Governor only for **clear workload separation**
- Always test classifier logic in non-production
- Document application names used for routing

4. When to Use Each Approach

Scenario	Recommended Method
General system tuning	Instance-level MAXDOP
Mixed OLTP + reporting	Resource Governor
One-off query tuning	OPTION (MAXDOP n)
Vendor application	Resource Governor (safer)