FLEXEra

FlexNet Operations 2018 Release 1

Recommendations for FlexNet Operations Performance Improvement

Legal Information

Book Name:	Recommendations for FlexNet Operations Performance Improvement
Part Number:	FNO-2018R1-RPI00
Product Release Date	February 2018
Last Modified Date	22 January 2018

Copyright Notice

Intellectual Property

Restricted Rights Legend

Contents

Contents

Recommendations for FlexNet Operations Performance Improvement

Introduction

Over time, a FlexNet Operations Microsoft SQL Server database can grow very large, degrading performance during operations such as activating licenses and loading landing pages. While it is reasonable to expect queries to take longer when there are millions of records in the database, actions can be taken to improve FlexNet Operations performance. Most of these actions are implemented on the database side by the database administrator, while some are implemented through the FlexNet Operations Producer Portal.

Performance issues can be caused by a lack of indexing on certain tables, depending on the volume of data and the types of operations. Because each instance of FlexNet Operations is different and requirements cannot be predicted in advance, indexes cannot be created before installation. The recommendations provided in the Database Index Creation section are based on use case and volume of data.

As the volume of data increases, the database administrator must perform certain operations to maintain reasonable performance. The General Recommendations section lists actions that the database administrator can take to improve FlexNet Operations performance.

The recommendations in this document are based on the assumption that performance degradation is caused by the database, not by the FlexNet Operations instance. If the degradation is caused by the FlexNet Operations instance, then the solution is to connect a cluster of FlexNet Operations nodes to the database and apply these recommendations.

Database Index Creation

In general, index creation improves performance in the use cases listed, but problematic queries should be analyzed case by case for other issues. For example, if an operation takes a long time, capture the SQL from the FlexNet Operations logs to get the query's execution plan. This plan should indicate any missing indexes and provide details about recommended ones. After the indexes are created, performance should improve to the best possible, considering the volume of data involved.

The indexes listed here were captured in this manner from various use cases:

- Device Landing Page

- Delete Entities in FlexNet Operations

- Recovery of Served Clients

Device Landing Page

```
CREATE NONCLUSTERED INDEX [NonClusteredIndex-20130314-165607] ON [dbo].[OPS_HOST_TYPES]
(
    [ID] ASC
)
INCLUDE ([NAME]) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
```

Delete Entities in FlexNet Operations

```
CREATE NONCLUSTERED INDEX [OPS_IDX_ACT_ITEM_XFER_LI]
    ON [dbo].[OPS_ACTIVATABLE_ITEM]
([TRANSFERRED_FROM_LINE_ITEM], [ACTIVATABLE_ITEM_TYPE])
    INCLUDE ([ID])

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[OPS_ACTIVATABLE_ITEM] ([SPLIT_FROM_LINE_ITEM], [ACTIVATABLE_ITEM_TYPE])
INCLUDE ([ID])

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
    ON [dbo].[OPS_ENTITLEMENT_ORDER] ([TRANSFERRED_FROM_ENT],[ENTITLEMENT_TYPE_DISCRIMINATOR])
    INCLUDE ([ID])

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[OPS_ACTIVATABLE_ITEM] ([SPLIT_FROM_LINE_ITEM])

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[OPS_ENTITLEMENT_ORDER] ([TRANSFERRED_FROM_ENT])

G
```

Recovery of Served Clients

```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[EMB_DEVICE] ([DEVICE_TYPE])
INCLUDE ([HOST], [SERVED_STATUS])

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[EMB_DEVICE_FEATURE] ([DEVICE_ID])

GO
```

General Recommendations

This section describes actions that the database administrator can take to improve FlexNet Operations performance:

- Clean Up OPS_REQUEST_TRANSACTION Table

- Update Database Statistics

- Set Appropriate Database Isolation Level

- Disable Allow Adding Redundant Server Setting

Clean Up OPS_REQUEST_TRANSACTION Table

After backing up the database, periodically clean up the OPS_REQUEST_TRANSACTION table, which holds all trusted activation transactions. This table tends to grow very large for customers who do a large number of trusted activations and contributes to FlexNet Operations startup delay. It is recommended that this table be monitored and older records removed to improve performance.

Update Database Statistics

Keep the database statistics up to date so queries work efficiently (MS SQLServer). To determine if the database is maintained in good condition for table scans and optimal performance, use the following SQL to evaluate how many days have elapsed since the last refresh for each table. It is recommended that the number of elapsed days be kept to less than a week for the most frequently used tables.

```
SELECT OBJECT_NAME(A.object_id) AS Object_Name,
    A.name AS index_name,
    STATS_DATE(A.OBJECT_ID, index_id) AS StatsUpdated ,
    DATEDIFF(d,STATS_DATE(A.OBJECT_ID, index_id),getdate()) DaysOld
FROM sys.indexes A
    INNER JOIN sys.tables B ON A.object_id = B.object_id
WHERE A.name IS NOT NULL
ORDER BY DATEDIFF(d,STATS_DATE(A.OBJECT_ID, index_id),getdate()) DESC
```

To update statistics on all the tables, run the following command:

```
EXEC sp_updatestats
```

Set Appropriate Database Isolation Level

Set the appropriate database isolation level—the degree to which an application process is isolated from other concurrently executing application processes—to maintain optimal performance. For SQLServer, the default isolation level is READ_COMMITTED, which means that tables are locked even during READ operations. This can cause database contention, especially when there are several concurrent requests resulting in optimistic lock exceptions and operations failures. For better performance in such cases, set the FlexNet Operations database isolation level to READ_COMMITTED_SNAPSHOT with this command:

```
alter database <database name> set READ_COMMITTED_SNAPSHOT on
```

Disable Allow Adding Redundant Server Setting

To improve FlexNet Embedded device activation performance, disable the **Allow Adding Redundant Server** setting if it is not required. This can be done through the FlexNet Operations Producer Portal.

Task   ***To disable the Allow Adding Redundant Server Setting***

1. Log in to the Producer Portal and navigate to **System Administration**.

2. Click **Embedded Devices** in the left panel.

3. Un-check the **Allow Adding Redundant Server** box in the right panel.