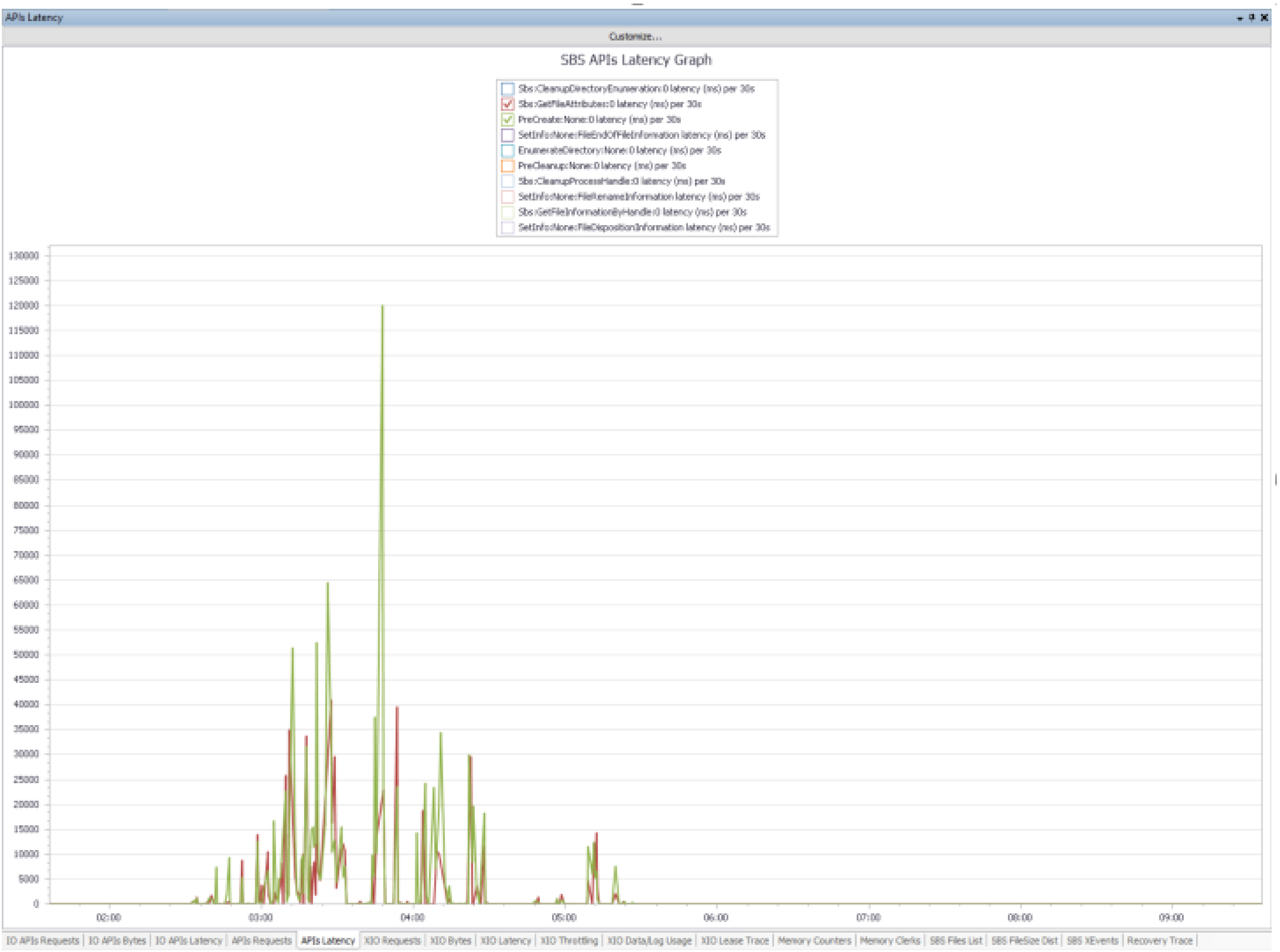
Verify SBS NSO locking contention

Last updated by | Abhishek Reddy Kumbham | Jun 15, 2021 at 1:01 PM PDT

Steps to Capture SBS Lock Times:

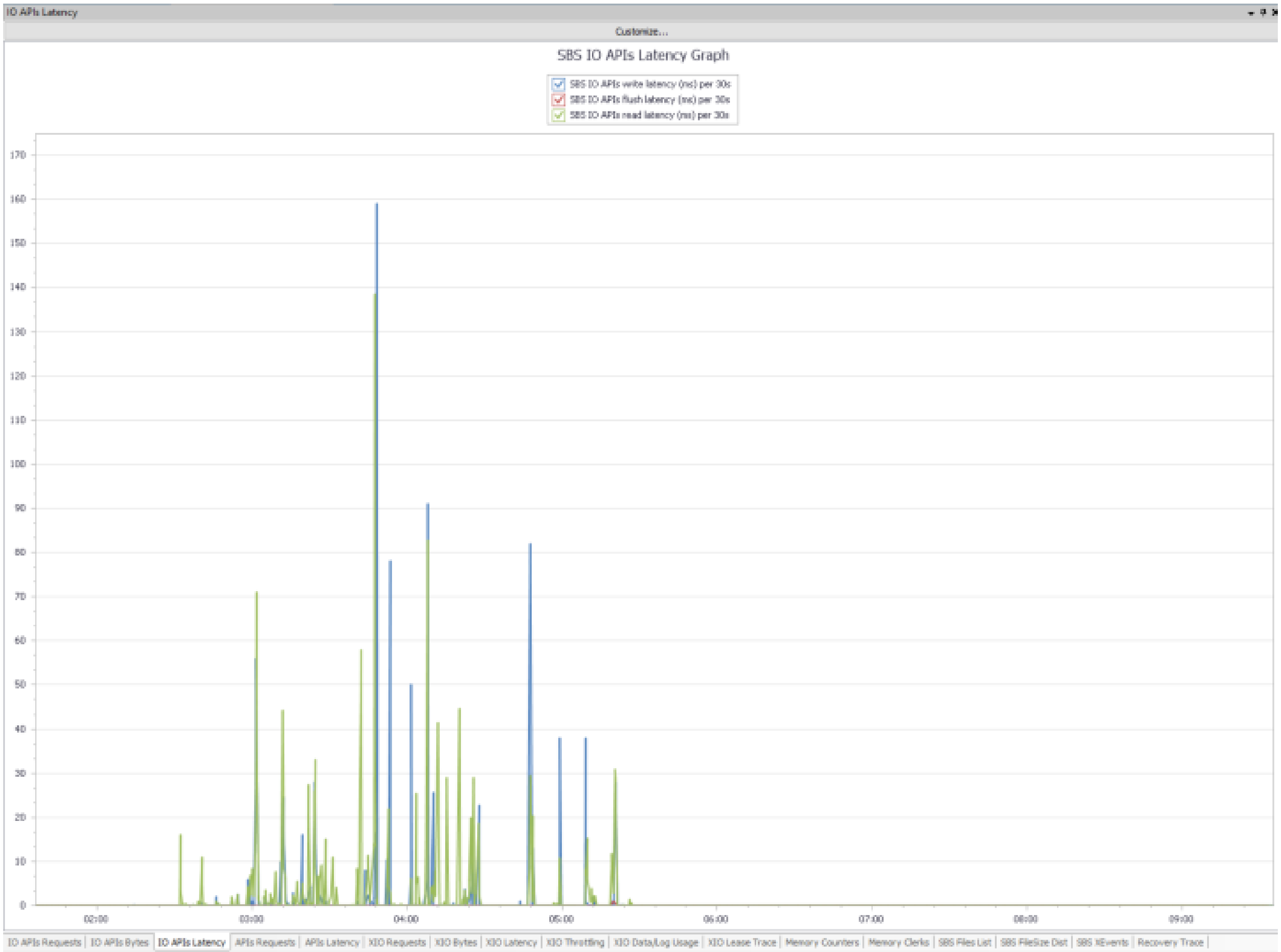
For big SKU PG SBS server, sometimes it may hit SBS NSO locking contention issue and customer may complain the query performance. To verify it's caused by this issue:

1. Open "SBS IO Overview.xts", jump to "APIs Latency" tab, you should see "Sbs:GetFileAttributes" and "PreCreate" have big latency (> 10000ms) when the issue happened.


APIs Latency
Customize...
SBS APIs Latency Graph
□ Sbs:CleanupDirectoryEnumeration:0 latency (ms) per 30s
☑ Sbs:GetFileAttributes:0 latency (ms) per 30s
☑ PreCreate:None:0 latency (ms) per 30s
□ SetInfo:None:FileEndOfFileInformation latency (ms) per 30s
□ EnumerateDirectory:None:0 latency (ms) per 30s
□ PreCleanup:None:0 latency (ms) per 30s
□ Sbs:CleanupProcessHandle:0 latency (ms) per 30s
□ SetInfo:None:FileRenameInformation latency (ms) per 30s
□ Sbs:GetFileInformationByHandle:0 latency (ms) per 30s
□ SetInfo:None:FileDispositionInformation latency (ms) per 30s
130000
125000
120000
115000
110000
105000
100000
95000
90000
85000
80000
75000
70000
65000
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
IO APIs Requests
IO APIs Bytes
IO APIs Latency
APIs Requests
APIs Latency
XIO Requests
XIO Bytes
XIO Latency
XIO Throttling
XIO Data/Log Usage
XIO Lease Trace
Memory Counters
Memory Clerks
SBS Files List
SBS FileSize Dist
SBS XEvents
Recovery Trace

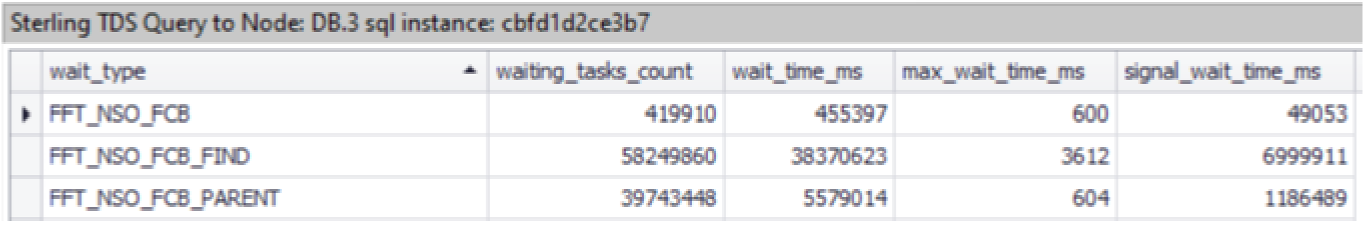
2. Check other types in this graph, they shouldn't have this issue. Though sometimes they might have few spikes, but most of the peak values should be < 5000ms.

3. Jump to "IO APIs Latency" tab, read/write/flush latency should very small (< 1ms). They might have spikes as well, but most of the peak values should be < 500ms.



4. Then open SBS Adhoc XTS view, execute: **select * from sys.dm_os_wait_stats where wait_type like 'FFT_NSO_FCB%'**

Then you should take a snapshot of these 3 rows:

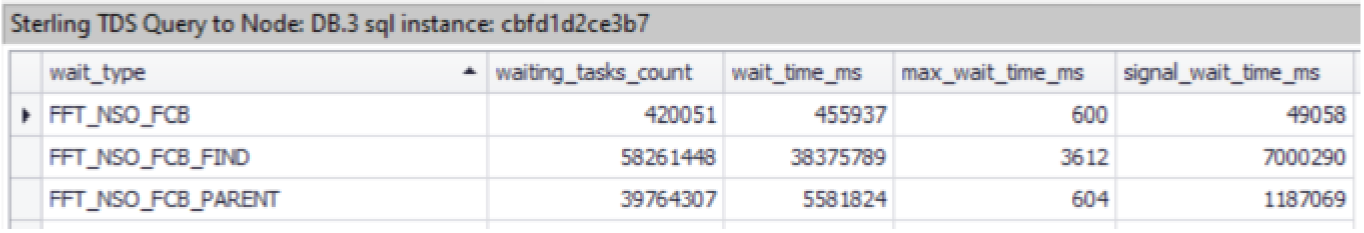


Sterling TDS Query to Node: DB.3 sql instance: cbfd1d2ce3b7

	wait_type ▲	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
▸	FFT_NSO_FCB	419910	455397	600	49053
	FFT_NSO_FCB_FIND	58249860	38370623	3612	6999911
	FFT_NSO_FCB_PARENT	39743448	5579014	604	1186489

These values are aggregated values from the server started. Unluckily we don't have history data of this table in Kusto. So we need to take snapshots and calculate the diff.

Wait about 2 minutes, run the query again and take the second snapshot:



Sterling TDS Query to Node: DB.3 sql instance: cbfd1d2ce3b7

wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
FFT_NSO_FCB	420051	455937	600	49058
FFT_NSO_FCB_FIND	58261448	38375789	3612	7000290
FFT_NSO_FCB_PARENT	39764307	5581824	604	1187069

While the issue is happening, you may want to capture a few more snapshots a couple more times.

Once captured, then you can calculate the **average wait_time_ms** for each task of each wait_type, between consecutive snapshots.

```
FFT_NSO_FCB
(455937 - 455397) / (420051 - 419910) = 3.8298

FFT_NSO_FCB_FIND
(38375789 - 38370623) / (58261448 - 58249860) = 0.4458

FFT_NSO_FCB_PARENT
(5581824 - 5579014) / (39764307 - 39743448) = 0.1347
```

In this example, these values are very small. But if you see any of these avg values > 3000ms, it means in this 2 minutes interval, the NSO locking contention is very severe.

PS: If you don't have a chance to know in advance when the performance issue, if the server hasn't restarted after the performance issue happened, then after you take the first snapshot, you can calculate the avg wait_time_ms directly to see if it's already big. If it is so it can prove the issue. If it's not, it cannot be used as evidence.

For example, to calculate FFT_NSO_FCB_FIND in the first snapshot: 38375789 / 58261448 = 0.6587