Root Cause Analysis

SQL Server Memory Crashes - WIN-O2GDS6TEBOI

Server	WIN-O2GDS6TEBOI (Azure Arc-enabled)
SQL Instances	MSSQLSERVER (default, TCP 1433) + DEMOSQLSERVER (named, dynamic port)
SQL Version	SQL Server 2019 Developer (15.0.2000.5)
Subscription	-
Investigation Period	Aug 19 - Aug 27, 2026
Data Source	Log Analytics (Perf, Event tables)
Report Date	August 27, 2026

1. Executive Summary

The server WIN-O2GDS6TEBOI is **severely memory-constrained**, running two SQL Server 2019 instances plus SSMS and multiple heavyweight agents on approximately **4 GB total RAM**. Two confirmed crash events were identified in the last 7 days, both following the same pattern: **memory exhaustion -> CPU thrashing (page fault I/O) -> process crash/kill -> memory release**. The root cause is unbounded SQL Server memory consumption (default max server memory) combined with an SSMS memory leak and heavy agent overhead.

Severity: HIGH Recurring crashes with data loss risk. Immediate remediation required.

2. Crash Event Timelines

2.1 Crash Event 1 - August 19, 2026 (~12:47 UTC)

Time (UTC)	Available Memory	Event
11:00 - 11:56	**45 - 180 MB**	Memory critically low, oscillating. System under extreme pressure.
12:00 - 12:04	**38 - 46 MB**	Near-zero available memory. OS struggling to allocate.
12:35	**28 MB**	Absolute minimum recorded - effectively OOM condition.
12:40	-	CPU hits **99.5%** - disk thrashing from page faults.
12:42	-	**SSMS.exe hangs** (AppHangTransient). WER event generated.
12:44	-	**EventID 17890** on both instances - SQL memory paged out. DEMOSQLSERVER working set: 48 MB (33%). MSSQLSERVER: 54 MB (35%).
12:47	**1,990 MB**	**CRASH/KILL** - Process terminated, ~1.9 GB freed instantly.
12:58	-	**RADAR_PRE_LEAK_64** - Windows confirms **memory leak in SSMS.exe**.
12:48 - 13:06	1,500 - 1,600 MB	Repeated EventID 17890: SQL memory still paged out post-crash.

2.2 Crash Event 2 - August 25, 2026 (~05:25 UTC)

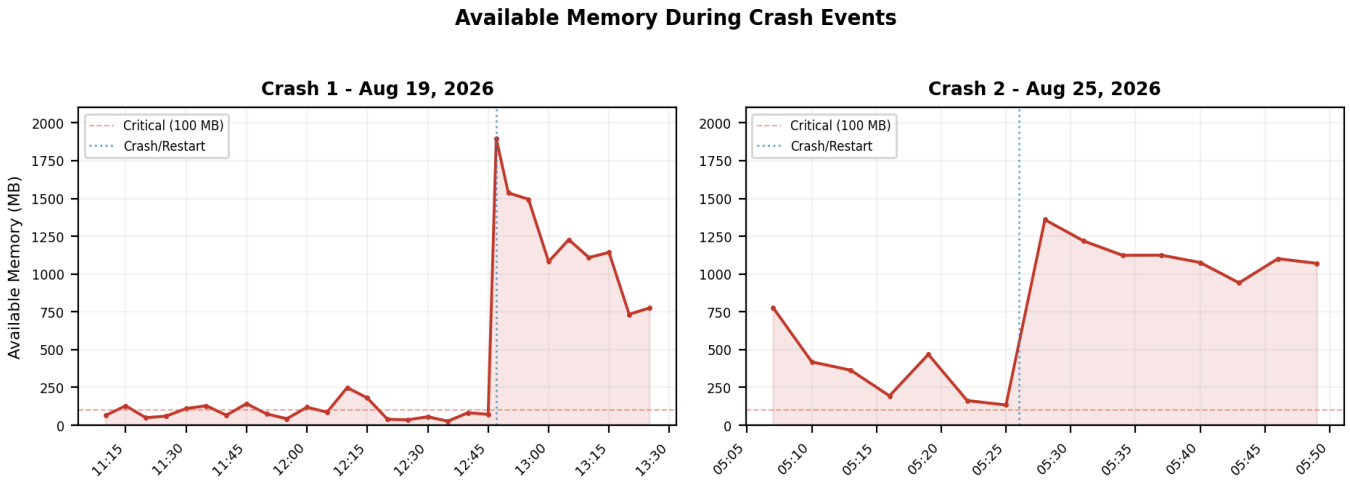
Time (UTC)	Available Memory	Event
05:05	-	CPU pegged at **100%** (sustained).
05:07 - 05:25	**140 - 816 MB**	Rapid memory consumption, declining sharply.

Time (UTC)	Available Memory	Event
05:25	**140 MB**	Minimum before crash. Same pattern as Crash 1.
05:26	**1,294 MB**	**CRASH/KILL** - Process terminated, memory released instantly.
05:30+	1,100 - 1,350 MB	Memory stabilizes post-restart. CPU drops to ~10-16%.

3. Telemetry Evidence

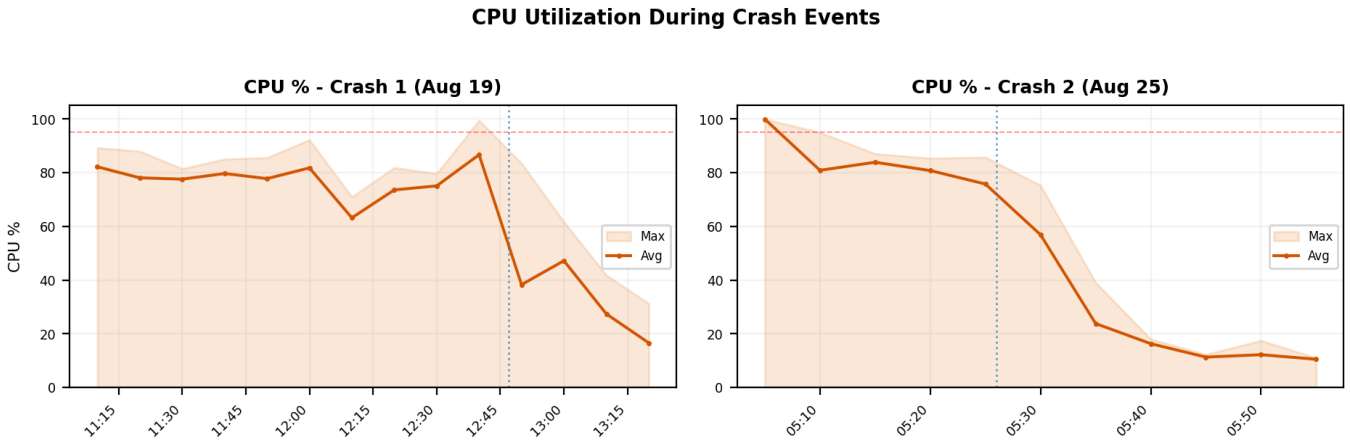
3.1 Memory Available Bytes

Both events show memory dropping to critically low levels (28-140 MB), followed by an instantaneous jump to 1.3-2.0 GB when the offending process is killed or crashes. The sawtooth pattern indicates repeated memory pressure cycles.



3.2 CPU Utilization

CPU spikes to 99-100% are not compute-bound workloads but disk I/O thrashing from aggressive page faulting. Once the crash occurs, CPU immediately drops to normal levels, confirming the correlation between memory exhaustion and CPU saturation.



4. Root Cause Analysis

4.1 Primary Cause: Unbounded SQL Server Memory

Both SQL Server instances (MSSQLSERVER and DEMOSQLSERVER) are configured with the **default max server memory setting (2,147,483,647 MB)**, meaning each instance will greedily claim all available system RAM. On a server with only ~4 GB total physical memory, this guarantees memory starvation within minutes of both instances starting up.

4.2 Aggravating Factors

- **SSMS.exe Memory Leak** - Confirmed by Windows Error Reporting RADAR_PRE_LEAK_64 event (Aug 19 12:58 UTC). SSMS was actively leaking memory and eventually hung (AppHangTransient), consuming RAM that SQL Server needed for buffer pool operations.
- **Heavy Agent Overhead** - Six concurrent agents compete for limited RAM: Azure Monitor Agent (AMA), Windows Admin Center (WAC - restarts every ~15 min), SQL Server Extension Agent, Microsoft Defender for Endpoint (SenseIR.exe), Azure Connected Machine Agent, and Change Tracking.
- **No Lock Pages in Memory** - EventID 17890 shows Windows aggressively paging SQL Server's working set to disk (down to 33% of committed memory), confirming the SQL Server service account lacks the 'Lock Pages in Memory' privilege.
- **CPU Thrashing from Page Faults** - The 99-100% CPU spikes are not computational load but disk I/O thrashing as the OS frantically swaps memory pages in and out of the page file. This cascading effect makes the entire server unresponsive.

4.3 Failure Chain



5. Key Evidence Summary

Evidence	Source	Finding
EventID 17890	MSSQL$DEMOSQLSERVER, MSSQLSERVER	"A significant part of sql server process memory has been paged out." Working set collapsed to 33-50% of committed. Repeated 10+ times.
RADAR_PRE_LEAK_64	Windows Error Reporting	Memory leak detected in SSMS.exe v22.6.11806.211. Process faulted.
AppHangTransient	Windows Error Reporting	SSMS.exe became unresponsive at 12:42 UTC due to memory pressure.
Available Bytes min	Perf counters (AMA)	Dropped to **28 MB** (Aug 19) and **140 MB** (Aug 25). Server has ~4 GB total.
CPU % max	Perf counters (AMA)	Hit **99.5%** (Aug 19) and **100%** (Aug 25) during memory pressure.
EventID 17177	MSSQLSERVER, DEMOSQLSERVER	Both instances running since Aug 19 10:26 UTC (PID 2828, 2812). No SQL restart between crashes.

6. Recommendations

Priority	Action	Details	Impact
P0	**Cap max server memory**	Set ~800 MB per instance: *EXEC sp_configure 'max server memory', 800; RECONFIGURE;*	Prevents unbounded consumption. Immediate fix.

Priority	Action	Details	Impact
P0	**Close SSMS when idle**	SSMS has confirmed memory leak (RADAR_PRE_LEAK_64). Use sqlcmd or Azure Data Studio instead.	Eliminates the leak that triggered Crash 1.
P1	**Increase VM RAM to 8+ GB**	Hyper-V host (Standard_D16s_v4) has capacity. Adjust the guest VM allocation.	Provides adequate headroom for 2 instances + agents.
P1	**Grant Lock Pages in Memory**	Add the SQL Server service account to the local policy "Lock Pages in Memory".	Prevents OS from paging out SQL buffer pool.
P2	**Consolidate to 1 instance**	Remove DEMOSQLSERVER if not actively needed for workloads.	Halves SQL memory footprint.
P2	**Reduce WAC polling**	Windows Admin Center restarts/reconfigures every ~15 min. Reduce or disable if not needed.	Frees CPU and memory cycles.