Laborübung 7 - Profiling, Code Analysing & Benchmarking

Code Hot Paths

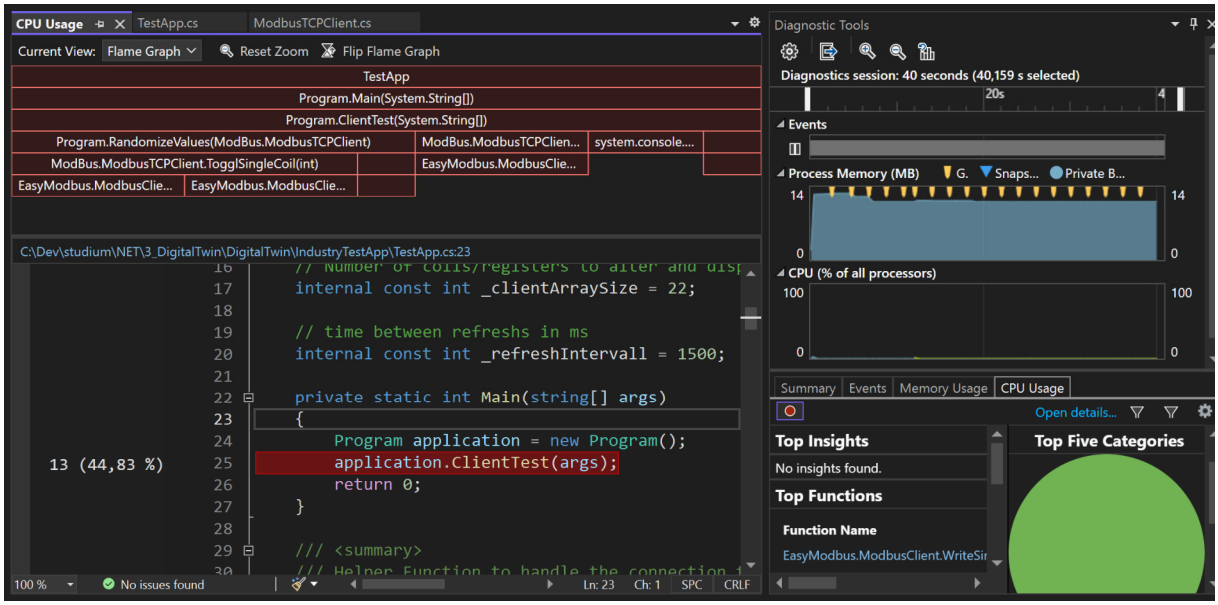


Abbildung 1: Code Hot Paths Debugging

Über das CPU Profiling ist erkennbar das die „Randomize.Values“ Funktion am meisten CPU-Zeit beansprucht. Das ist nicht weiter verwunderlich, nachdem diese Funktion in unserem Run-Loop für jeden Produktions-Zyklus aufgerufen wird.

Memory Profiling

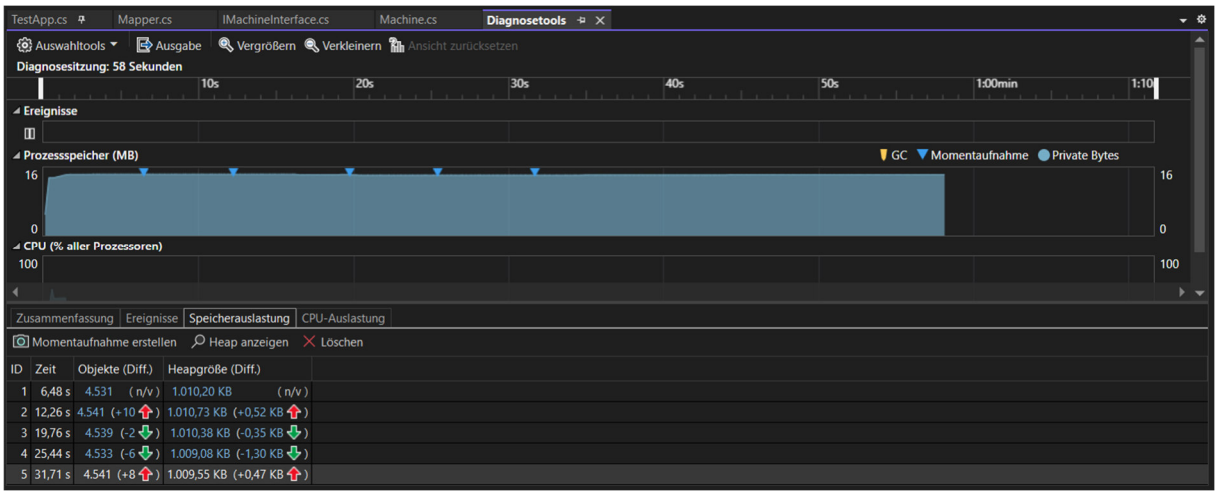


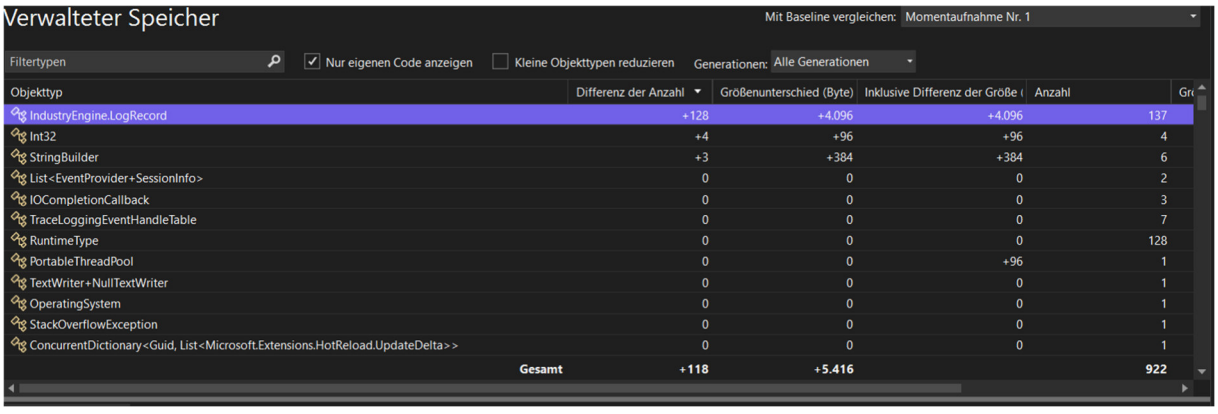
Abbildung 2: Memory Profiling der DigitalTwin Anwendung

Wie in Abbildung 1 zu sehen, variieren die Anzahl der Objekte und die Heapgröße während der Ausführung geringfügig. Wir lassen das Programm noch einige Zeit laufen und machen noch einen Snapshot. Wie in Abbildung 3 zu sehen, ist die Anzahl der Objekte sehr viel größer.

ID	Zeit	Objekte (Diff.)	Heapgröße (Diff.)
1	6,48 s	4.531 (n/v)	1.010,20 KB (n/v)
2	12,26 s	4.541 (+10 ↑)	1.010,73 KB (+0,52 KB ↑)
3	19,76 s	4.539 (-2 ↓)	1.010,38 KB (-0,35 KB ↓)
4	25,44 s	4.533 (-6 ↓)	1.009,08 KB (-1,30 KB ↓)
5	31,71 s	4.541 (+8 ↑)	1.009,55 KB (+0,47 KB ↑)
6	111,91 s	4.644 (+103 ↑)	1.014,38 KB (+4,84 KB ↑)

Abbildung 3: Memory Profiling mit mehreren Snapshots

Um einen besseren Einblick zu bekommen, vergleichen wir den ersten mit dem letzten Snapshot. Wie folgende Abbildung zeigt, wird während der Laufzeit die Anzahl der LogRecord Objekte stetig größer.



Objekttyp	Differenz der Anzahl	Größenunterschied (Byte)	Inklusive Differenz der Größe	Anzahl	Gr
IndustryEngine.LogRecord	+128	+4.096	+4.096	137	
Int32	+4	+96	+96	4	
StringBuilder	+3	+384	+384	6	
List<EventProvider+SessionInfo>	0	0	0	2	
IOCompletionCallback	0	0	0	3	
TraceLoggingEventHandleTable	0	0	0	7	
RuntimeType	0	0	0	128	
PortableThreadPool	0	0	+96	1	
TextWriter+NullTextWriter	0	0	0	1	
OperatingSystem	0	0	0	1	
StackOverflowException	0	0	0	1	
ConcurrentDictionary<Guid, List<Microsoft.Extensions.HotReload.UpdateDelta>>	0	0	0	1	
Gesamt	+118	+5.416		922	

Abbildung 4: Memory Profiling Snapshot-Vergleich

Auch das ist zu erwarten, nachdem wir diese als List<LogRecord> für unser Maschin-Logging verwendet wird.

Auf einem Microsoft Surface Gerät steht mit 8GB nur wenig Speicher zur Verfügung. Vermutlich kommt es dadurch im Vergleich zum Lenovo Gerät in Abbildung 2, häufiger zu GC Läufen, wie in folgender Darstellung ersichtlich:

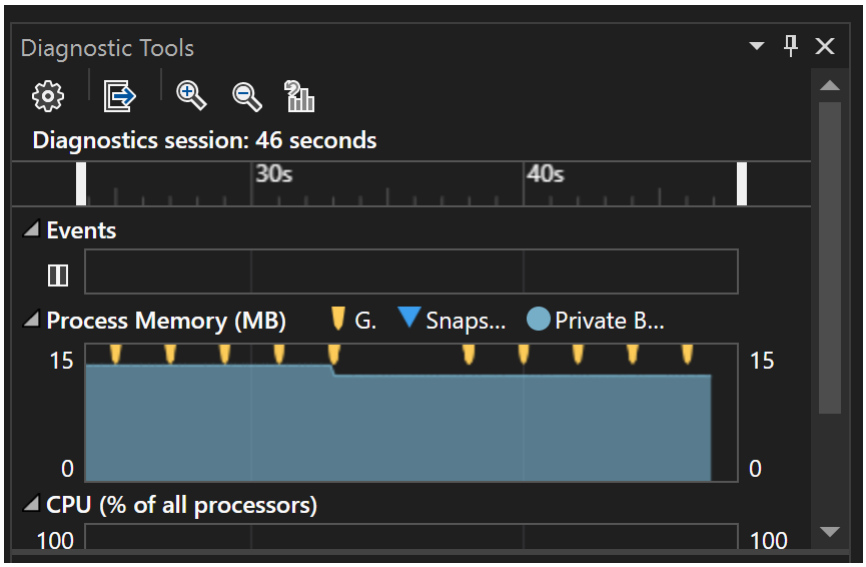
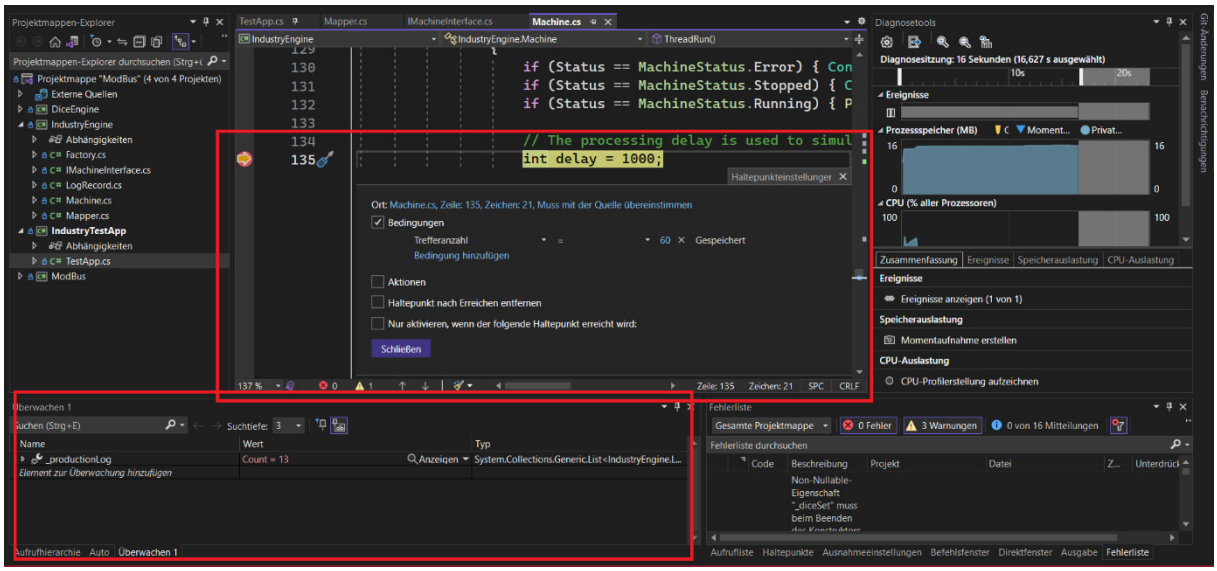


Abbildung 5: Memory Profiling bei hoher Speicherbelegung



In unserer klasse Maschine werden Aktivitäten geloggt. Mit dem Debugger ist es möglich mit bedingten Breakepoint an eine gewisse stelle springen zu lassen.