DebugDiag Analysis Report

Dumps: 1/1

Rules: 1/1

0
Error
1
Warning
1
Information
2
Notification

Analysis Summary

Warning

Description	Recommendation
Number of objects ready for finalization: 6059	This is an indication that your finalizer thread may be blocked. Look at finalizequeue info and finalizer stack to determine why/if the finalizer is blocked

Information

Description	Recommendation
GC Heap usage: 3.24 GBytes Common issues for high .NET Memory usage include: Blocked finalizers, lots of memory in cache/sessions, lots of large objects and memory rooted in statics You should also revivew the most memory consuming objects.	

Notification

Description	Recommendation
One or more of the selected rules were not completed.	See the Analysis Rule Summary for more information.
The process is currently in the middle of a garbage collection	Any information about the GC heaps and the objects on the GC heaps may be invalid

Analysis Details

DotNetMemoryAnalysis

CLR Memory Analysis for SedonaOne.DMP

Type of Analysis Performed	**.Net Memory Pressure Analysis**
Machine Name	**RYZEN-DESKTOP**
Operating System	**Unexpected**
Number Of Processors	**16**
Process ID	**21424**
Process Image	**C:\Users\Travis\Documents\Visual Studio 2017\Projects\SedonaOneDev\SedonaOne\SedonaOne\bin\Debug\net461\win10-x64\SedonaOne.exe**
Command Line	**"C:\Users\Travis\Documents\Visual Studio 2017\Projects\SedonaOneDev\SedonaOne\SedonaOne\bin\Debug\net461\win10-x64\SedonaOne.exe"**
System Up-Time	**3 day(s) 20:39:14**
Process Up-Time	**00:05:07**
Processor Type	**X64**
Process Bitness	**64-Bit**

CLR Information

CLR version = **4.7.2110.0**
Microsoft.Diagnostics.Runtime version = **0.9.2.0**

.NET GC Heap Information

Number of GC Heaps: 16

Heap Size	0xa9900d8 (177,799,384)
Heap Size	0xdb1fd00 (229,768,448)
Heap Size	0xbce1e20 (198,057,504)
Heap Size	0xca0b000 (211,857,408)
Heap Size	0xb91aa58 (194,095,704)
Heap Size	0xcf0d038 (217,108,536)
Heap Size	0xc37e1a8 (204,988,840)
Heap Size	0xc145a90 (202,660,496)
Heap Size	0xd51a118 (223,453,464)
Heap Size	0xe31a730 (238,135,088)
Heap Size	0x10b40870 (280,234,096)
Heap Size	0xbedb930 (200,128,816)

Heap Size	0xda04720 (228,607,776)
Heap Size	0xc00d000 (201,379,840)
Heap Size	0xd9cf0e0 (228,389,088)
Heap Size	0xe439970 (239,311,216)

GC Heap Size	**3.24 GBytes**
Total Commit Size	**3316 MB**
Total Reserved Size	**17163 MB**

40 most memory consuming .NET object types

Free
System.Byte[]
System.Int32[]
System.Reflection.Emit.GenericMethodInfo
System.Object[]
System.Linq.Expressions.UnaryExpression
System.Collections.ObjectModel.ReadOnlyCollection<System.Linq.Expressions.Expression>
System.Collections.Generic.Dictionary+Entry<System.Linq.Expressions.ParameterExpression,System.Linq.Expressions.Compiler.VariableStorageKind>[]
System.Collections.Generic.Dictionary+Entry<System.Linq.Expressions.ParameterExpression,System.Int32>[]
System.Collections.Generic.Dictionary+Entry<System.Linq.Expressions.ParameterExpression,System.Linq.Expressions.Compiler.CompilerScope+Storage>[]
System.Linq.Expressions.InstanceMethodCallExpression2
System.Linq.Expressions.Compiler.CompilerScope
System.Linq.Expressions.Expression[]
System.Collections.Generic.Dictionary<System.Linq.Expressions.ParameterExpression,System.Linq.Expressions.Compiler.CompilerScope+Storage>
System.Collections.Generic.Dictionary<System.Linq.Expressions.ParameterExpression,System.Linq.Expressions.Compiler.VariableStorageKind>
System.Collections.Generic.Dictionary<System.Linq.Expressions.ParameterExpression,System.Int32>
System.Runtime.CompilerServices.TrueReadOnlyCollection<System.Linq.Expressions.Expression>
System.Linq.Expressions.ListArgumentProvider
System.RuntimeMethodHandle
System.Reflection.Emit.__FixupData[]
System.RuntimeTypeHandle
System.Linq.Expressions.NewExpression
System.Runtime.CompilerServices.TrueReadOnlyCollection<System.Linq.Expressions.ParameterExpression>
System.Linq.Expressions.ParameterExpression[]
System.Linq.Expressions.TypedConstantExpression
System.Linq.Expressions.BlockExpressionList
System.Linq.Expressions.AssignBinaryExpression
System.Linq.Expressions.Block2
System.Linq.Expressions.ConditionalExpression
System.Linq.Expressions.ScopeN
System.Linq.Expressions.Compiler.LabelScopeInfo
System.Collections.Generic.Dictionary+Entry<System.Object,System.Linq.Expressions.Compiler.CompilerScope>[]
System.Linq.Expressions.ConstantExpression
System.Linq.Expressions.ParameterExpression
System.Collections.Generic.Dictionary+ValueCollection<System.Linq.Expressions.ParameterExpression,System.Linq.Expressions.Compiler.CompilerScope+Storage>
System.String
System.Data.Entity.Core.Metadata.Edm.TypeUsage
System.Data.Entity.Core.Metadata.Edm.MetadataProperty
System.Collections.Generic.List<System.Linq.Expressions.Compiler.LambdaCompiler+WriteBack>
System.Collections.Generic.List+Enumerator<System.Linq.Expressions.Compiler.LambdaCompiler+WriteBack>

Color	Object type
Red	System.Web.UI... objects
Blue	System.Data... objects
Green	System.XML... objects
Purple	Custom objects

Top Objects in the Finalizer queue

As the number of finalizable objects is more than 0, please check the finalizer thread to see if it is blocked or active

Finalizer Thread
ntdll!NtWaitForMultipleObjects+14
KERNELBASE!WaitForMultipleObjectsEx+f0
clr!FinalizerThread::WaitForFinalizerEvent+b6
clr!FinalizerThread::FinalizerThreadWorker+54
clr!ManagedThreadBase_DispatchInner+39
clr!ManagedThreadBase_DispatchMiddle+6c

clr!ManagedThreadBase_DispatchOuter+75
[[DebuggerU2MCatchHandlerFrame]]
clr!FinalizerThread::FinalizerThreadStart+10a
clr!Thread::intermediateThreadProc+86
kernel32!BaseThreadInitThunk+14
ntdll!RtlUserThreadStart+21

Objects on the Large Object Heap

Free
System.Byte[]
System.Object[]
System.Reflection.Emit.__FixupData[]
System.Collections.Generic.Dictionary+Entry<System.Object,System.Linq.Expressions.Compiler.CompilerScope>[]
System.Int32[]
Roslyn.Utilities.StringTable+Entry[]
Roslyn.Utilities.TextKeyedCache+SharedEntry<Microsoft.CodeAnalysis.CSharp.Syntax.InternalSyntax.SyntaxTrivia>[]
Roslyn.Utilities.TextKeyedCache+SharedEntry<Microsoft.CodeAnalysis.CSharp.Syntax.InternalSyntax.SyntaxToken>[]
Microsoft.CodeAnalysis.Syntax.InternalSyntax.SyntaxNodeCache+Entry[]
System.Object[]
System.Collections.Generic.Dictionary+Entry<System.Data.Entity.Core.Metadata.Edm.EdmMember,System.Collections.Generic.KeyValuePair<System.Data.Entity.Core.M
System.Collections.Hashtable+bucket[]
System.Collections.Concurrent.ConcurrentDictionary+Node<System.Reflection.PropertyInfo,System.Collections.Generic.IEnumerable<System.Attribute>>[]
System.Collections.Generic.Dictionary+Entry<System.String,System.ValueTuple<System.Int32,System.Int32>>[]

More information:A high amount of large objects (strings and arrays over 85000 bytes) can lead to GC Heap framgmentation and thus higher memory usage in your application.
Look through the large objects, to dig deeper you can run !do on the object address in windbg, to see if these objects are expected and if you can minimize their usage in any way, by caching etc.
Common reasons for high amounts of large objects are large viewstate and Dataset serialization

Top 10 DataTables in the memory dump

Analysis Rule Summary

Rows	ColumnsCount	Columns	Size

Rule Name	Status	Details
DotNetMemoryAnalysis - *v (2.2.0.14)*	**Failed**	**Dump File:** C:\Users\Travis\AppData\Local\Temp\SedonaOne.DMP; **Type:** **System.InvalidOperationException** **Message:** Reading invalid pointer **Stack Trace:** DebugDiag.AnalysisRules.DotNetMemoryAnalysis.ObjRef.**Enqueue**(UInt64 addr, ClrHeap heap, ClrType possibleType, Queue`1 queue, HashSet`1 seenObjs) DebugDiag.AnalysisRules.DotNetMemoryAnalysis.ObjRef.**GetFieldReferences**(ClrHeap heap, Queue`1 queue, HashSet`1 seenObjs) DebugDiag.AnalysisRules.DotNetMemoryAnalysis.ObjSizeHelper.**ObjSize**(ClrHeap heap, ClrType objType, UInt64 objAddress) DebugDiag.AnalysisRules.DotNetMemoryAnalysis.**BangObjSize**(String obj) DebugDiag.AnalysisRules.DotNetMemoryAnalysis.**DumpDataTables**() DebugDiag.AnalysisRules.DotNetMemoryAnalysis.**DoDotNetMemoryAnalysis**() DebugDiag.AnalysisRules.DotNetMemoryAnalysis.**RunAnalysisRule**(NetScriptManager manager, NetProgress progress) DebugDiag.DotNet.NetAnalyzer.RunAnalysisRulesInternal(DumpFileType bitness, NetProgress progress, String symbolPath, String imagePath, String reportFileFullPath, Boolean twoTabs, AnalysisModes analysisMode)

Table of Contents