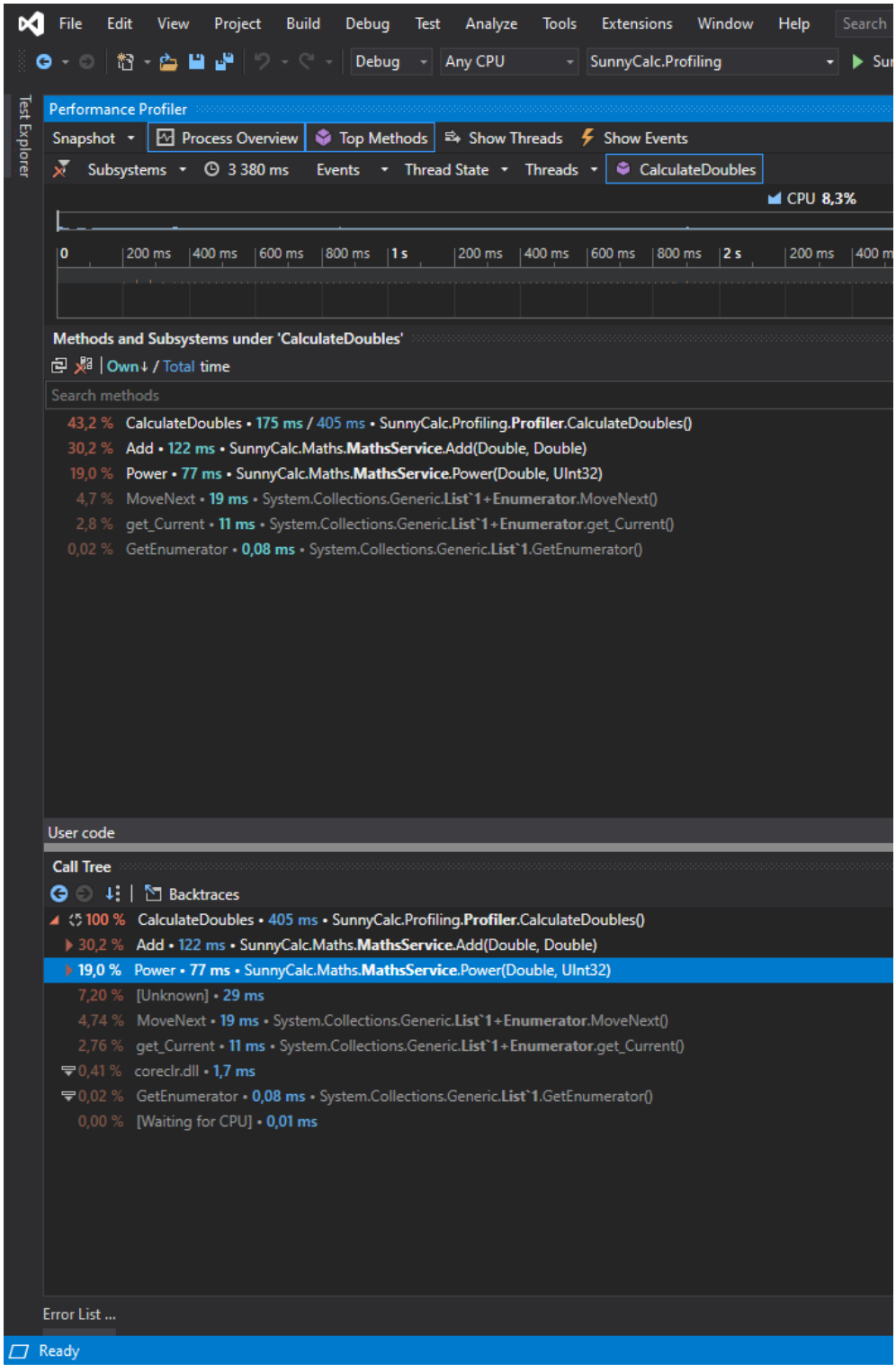
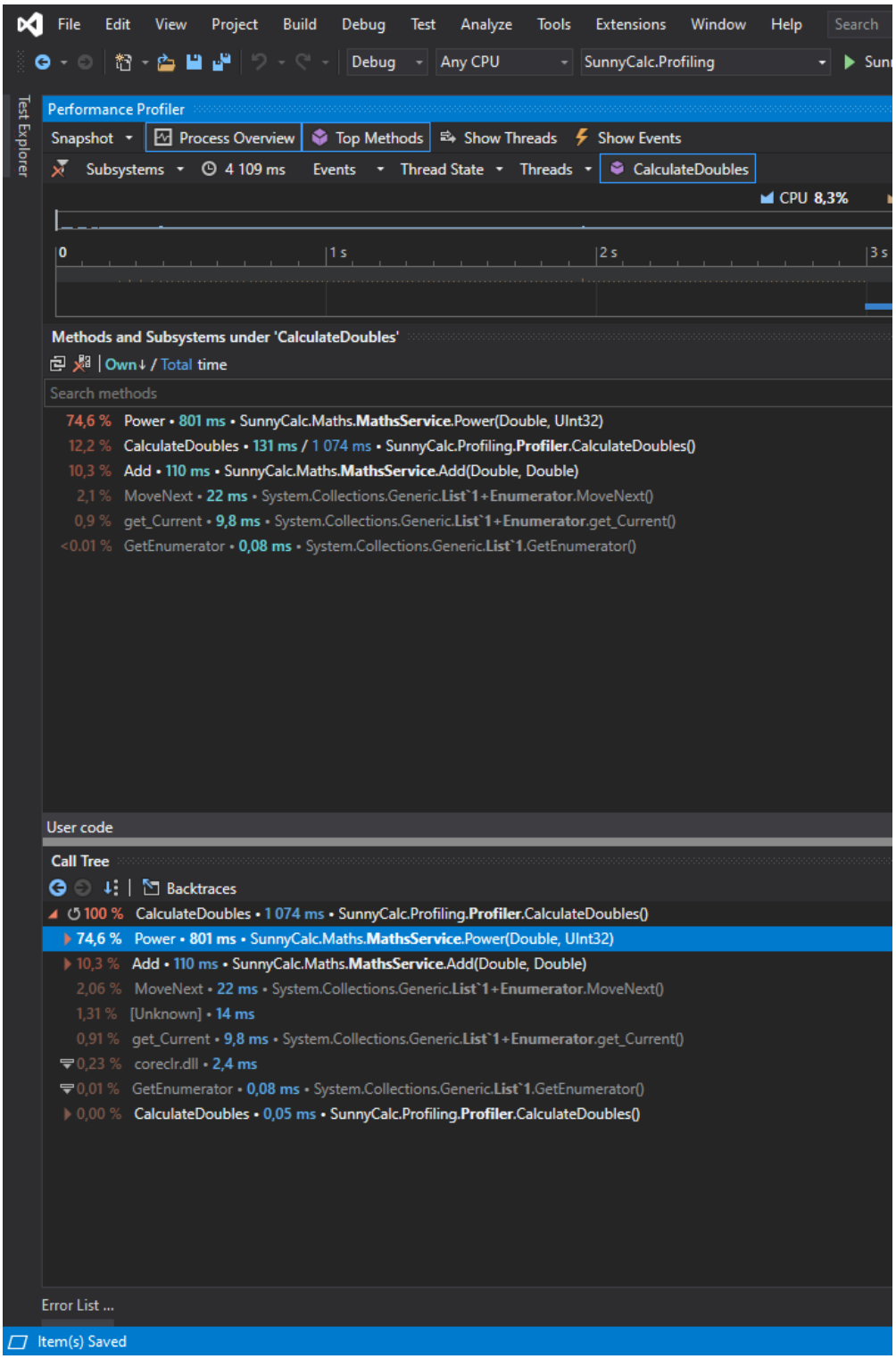
Protokol o profilingu

Profiling matematické knihovny byl proveden na výpočtu směrodatné odchylky z cca 14 000 000 náhodných čísel. Použita byla sada nástrojů JetBrains dotTrace. Přiloženy jsou výstupy profileru zobrazitelné příslušnými aplikacemi ze sady dotTrace.

Bylo zjištěno, že matematická knihovna je velmi efektivní. Samotný výpočet odchylky zabere při spuštěném profilování metodou Sampling asi 400 ms, nejvíce času (cca 25 % času běhu výpočetní metody `CalculateDoubles`, záleží na použité metodě profilingu, screenshot 1 níže zachycuje metodu Timeline) program tráví ve sčítací metodě Add, která už nemůže být dále optimalizována.



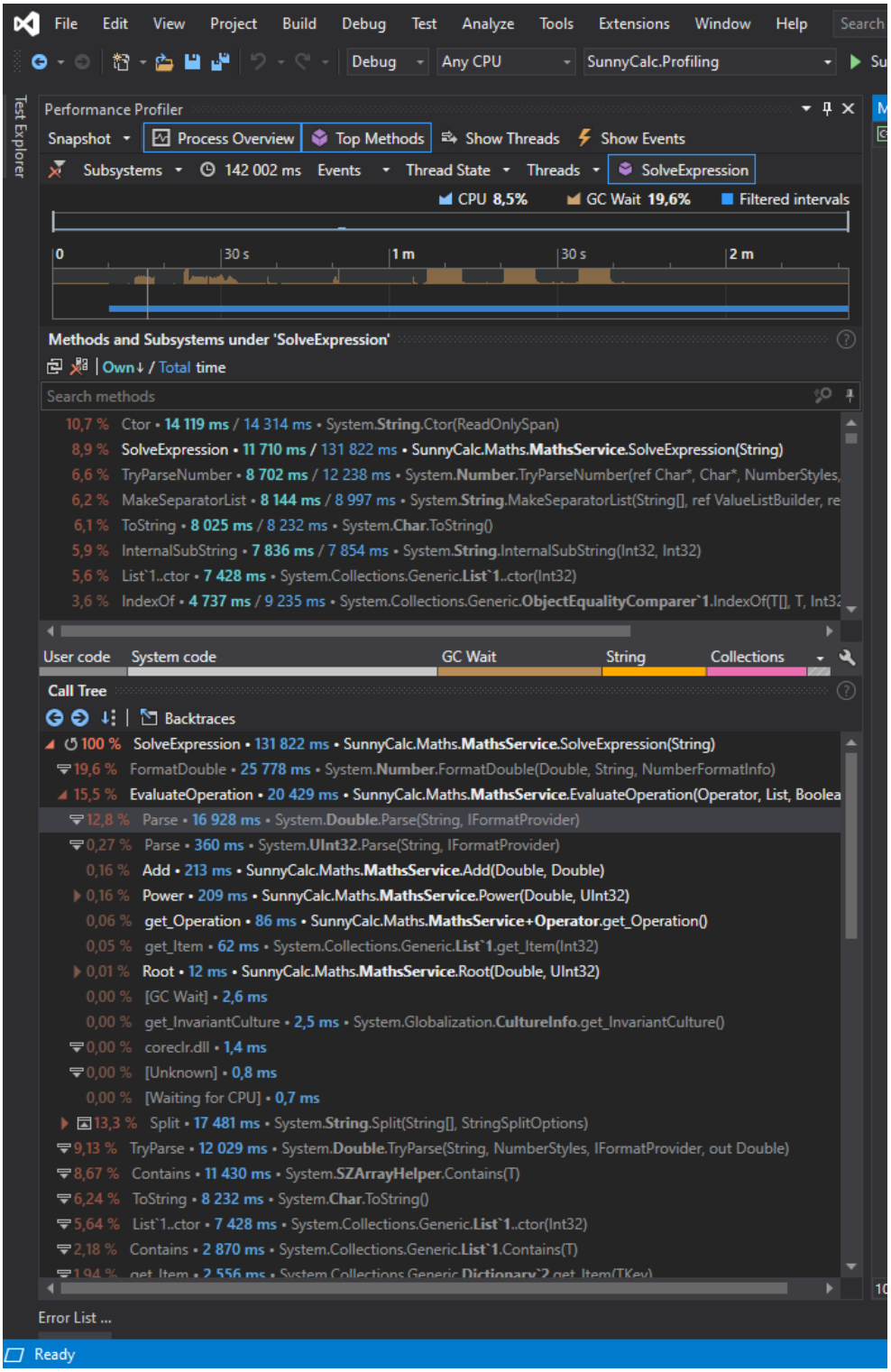
V umocňovací metodě `Power` program stráví asi 15 % času běhu výpočetní metody. Ukázalo se, že pokud by byl kód použitý v naší knihovně nahrazen voláním metody `Math.Pow` ze standardní knihovny .NET, výpočet by byl více než 10× pomalejší (viz screenshot 2).

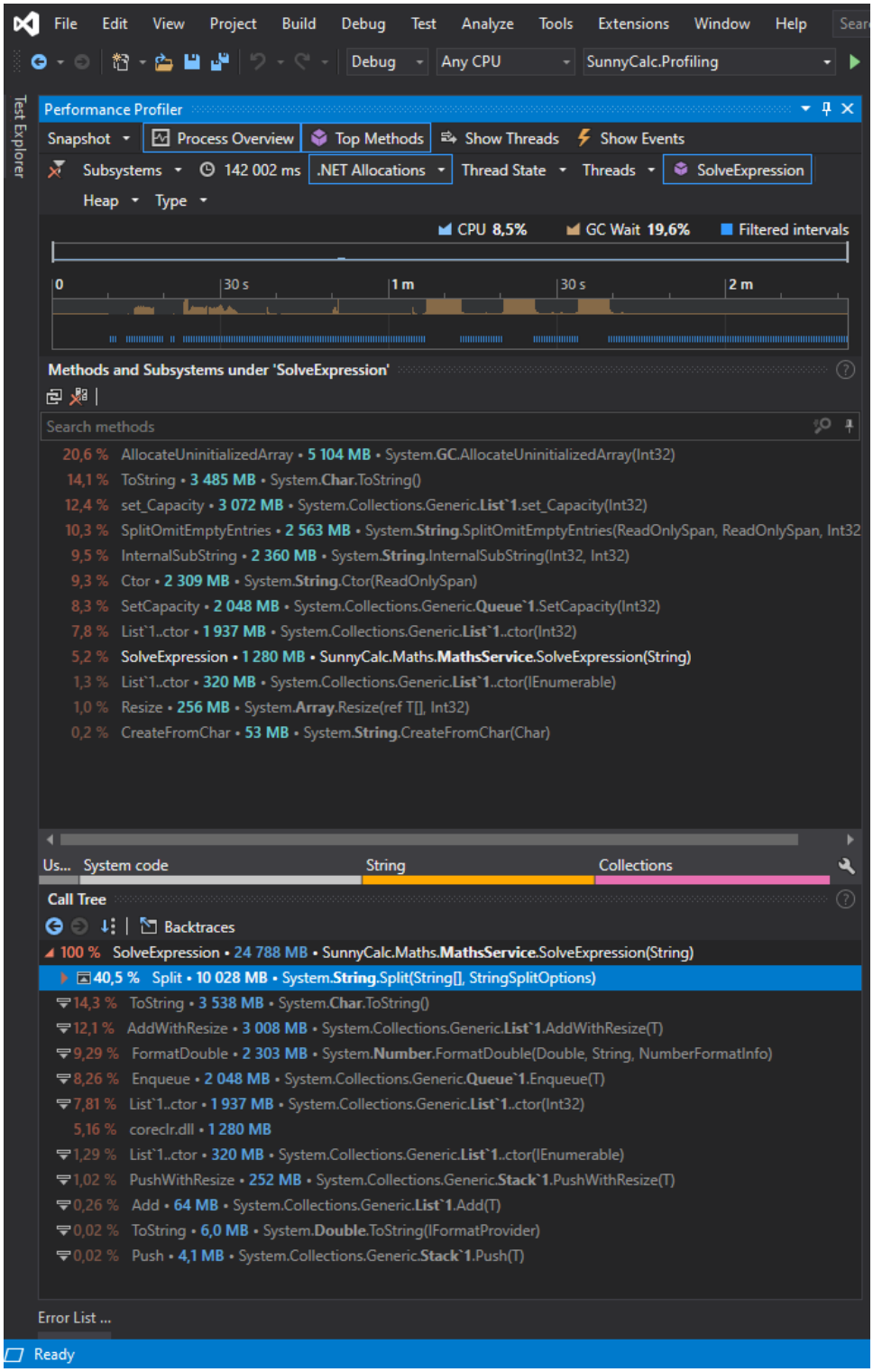


Profilingem byl změřen také výkon metody pro řešení komplexních matematických výrazů. Testovací metoda vytvořila ze vstupních čísel v paměti příslušný výraz, který byl předán metodě `SolveExpression`.

Při profilování bylo zjištěno, že zdaleka nejvíce náročné jsou operace s textovými řetězci: parsování řetězců na čísla, převádění čísel zpět na řetězce, vyhledávání v řetězcích a rozdělování řetězců. V těchto voláních tráví program až 45 % času. Částečná optimalizace používání těchto operací by byla možná, z principu se jim ale nelze vyhnout.

Zajímavým zjištěním byla paměťová náročnost. Při zpracování výrazu si metoda dohromady naalokuje téměř 25 GB paměti (která je průběžně uvolňována Garbage Collectorem). 40 % této paměti naalokuje metoda Split, která rozděluje řetězec na části podle použitých operátorů. I toto by mohlo být optimalizováno, pravděpodobně by to ale vyžadovalo zvolení jiného algoritmu.




File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Debug
Any CPU
SunnyCalc.Profiling
Test Explorer
Performance Profiler
Snapshot
Process Overview
Top Methods
Show Threads
Show Events
Subsystems
142 002 ms
.NET Allocations
Thread State
Threads
SolveExpression
Heap
Type
CPU 8,5%
GC Wait 19,6%
Filtered intervals
0
30 s
1 m
30 s
2 m
Methods and Subsystems under 'SolveExpression'
Search methods
20,6 % AllocateUninitializedArray • 5 104 MB • System.GC.AllocateUninitializedArray(Int32)
14,1 % ToString • 3 485 MB • System.Char.ToString()
12,4 % set_Capacity • 3 072 MB • System.Collections.Generic.List`1.set_Capacity(Int32)
10,3 % SplitOmitEmptyEntries • 2 563 MB • System.String.SplitOmitEmptyEntries(ReadOnlySpan, ReadOnlySpan, Int32
9,5 % InternalSubString • 2 360 MB • System.String.InternalSubString(Int32, Int32)
9,3 % Ctor • 2 309 MB • System.String.Ctor(ReadOnlySpan)
8,3 % SetCapacity • 2 048 MB • System.Collections.Generic.Queue`1.SetCapacity(Int32)
7,8 % List`1..ctor • 1 937 MB • System.Collections.Generic.List`1..ctor(Int32)
5,2 % SolveExpression • 1 280 MB • SunnyCalc.Maths.MathsService.SolveExpression(String)
1,3 % List`1..ctor • 320 MB • System.Collections.Generic.List`1..ctor(IEnumerable)
1,0 % Resize • 256 MB • System.Array.Resize(ref T[], Int32)
0,2 % CreateFromChar • 53 MB • System.String.CreateFromChar(Char)
Us...
System code
String
Collections
Call Tree
Backtraces
100 % SolveExpression • 24 788 MB • SunnyCalc.Maths.MathsService.SolveExpression(String)
40,5 % Split • 10 028 MB • System.String.Split(String[], StringSplitOptions)
14,3 % ToString • 3 538 MB • System.Char.ToString()
12,1 % AddWithResize • 3 008 MB • System.Collections.Generic.List`1.AddWithResize(T)
9,29 % FormatDouble • 2 303 MB • System.Number.FormatDouble(Double, String, NumberFormatInfo)
8,26 % Enqueue • 2 048 MB • System.Collections.Generic.Queue`1.Enqueue(T)
7,81 % List`1..ctor • 1 937 MB • System.Collections.Generic.List`1..ctor(Int32)
5,16 % coreclr.dll • 1 280 MB
1,29 % List`1..ctor • 320 MB • System.Collections.Generic.List`1..ctor(IEnumerable)
1,02 % PushWithResize • 252 MB • System.Collections.Generic.Stack`1.PushWithResize(T)
0,26 % Add • 64 MB • System.Collections.Generic.List`1.Add(T)
0,02 % ToString • 6,0 MB • System.Double.ToString(IFormatProvider)
0,02 % Push • 4,1 MB • System.Collections.Generic.Stack`1.Push(T)
Error List ...
Ready