

## Protokol o profilingu

Profiling matematické knihovny byl proveden na výpočtu směrodatné odchylky z 10, 100, 1000 a cca 14 000 000 náhodných čísel. Použita byla sada nástrojů JetBrains dotTrace. Přiloženy jsou některé výstupy profileru zobrazitelné příslušnými aplikacemi ze sady dotTrace.

Bylo zjištěno, že matematická knihovna je velmi efektivní. Při výpočtu z 10, 100 a 1000 nepřekročil čas strávený voláním metod matematické knihovny 1 ms. Časově nejnáročnější operací bylo počítání odmocniny, které zabralo asi 0,014 ms.

Properties of Root			
<b>Root(Double, UInt32)</b>			
Class:	SunnyCalc.Maths.MathsService		
Module:	SunnyCalc.Maths.dll (v1.0.0.0)		
MVID:	1D251E74-AD4B-4475-9515-4934E60DFA67		
	This call	In this tab	In snapshot
Number of calls	1	1	1
Time	0,014 ms (1%)	0,014 ms (1%)	0,014 ms (0%)
Own time	0,014 ms (1%)	0,014 ms (1%)	0,014 ms (0%)
Average time	0,014 ms	0,014 ms	0,014 ms
Instances	-	1	1

Pokusit se dále optimalizovat metodu Root je možné. Následují snímky demonstrující časovou náročnost volání ve výpočetní metodě pro 10, 100 a 1000 vstupních hodnot.

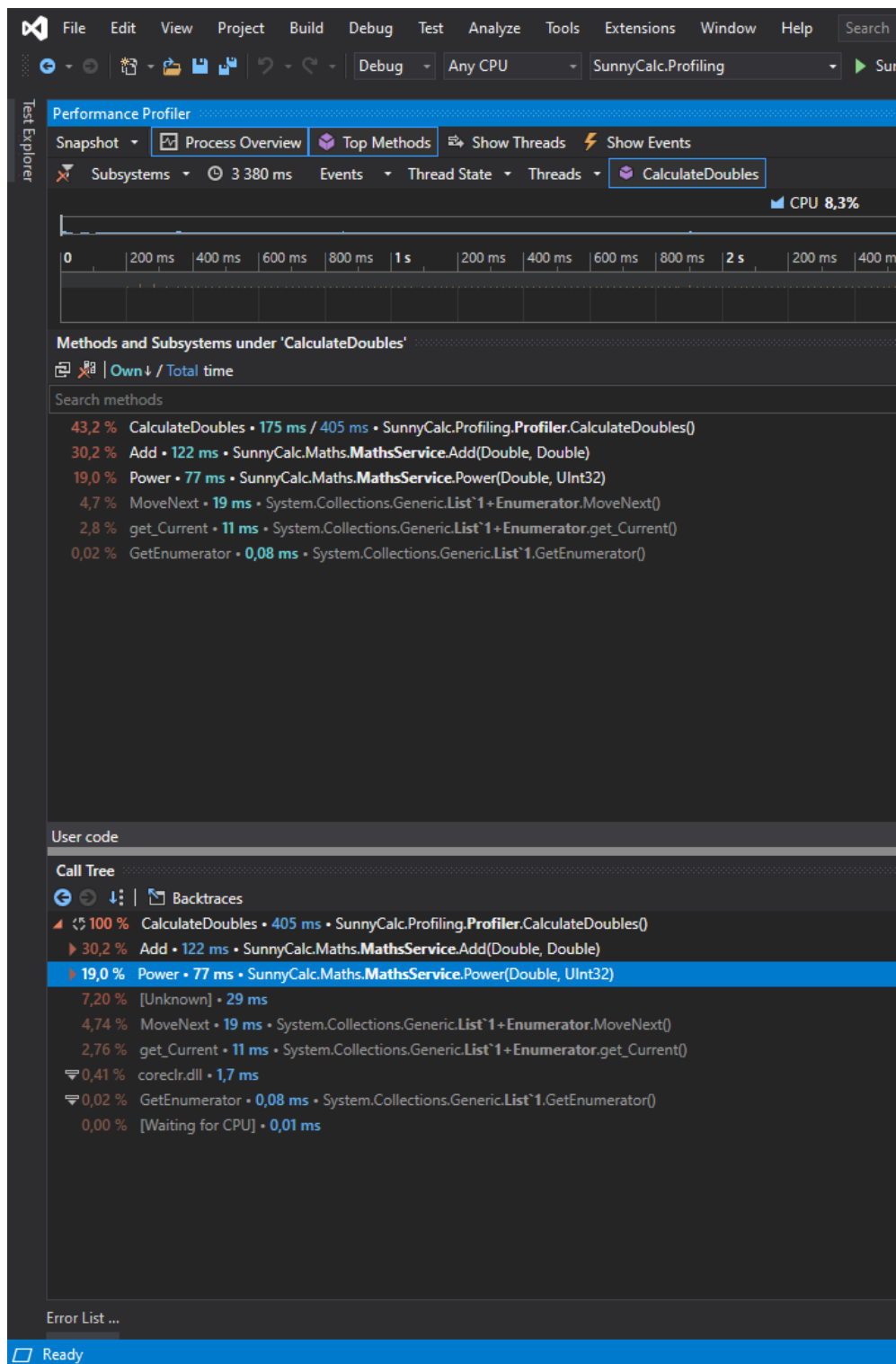
Overview		Profiler.CalculateDoubles X	
▲ 100,00 %	CalculateDoubles	• 1 ms • 1 call	• SunnyCalc.Profiling.Profiler.CalculateDoubles
▼ 11,14 %	MoveNext	• 0 ms • 11 calls	• System.Collections.Generic.List+Enumerator`1.MoveNext
1,74 %	Root	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Root(Double, UInt32)
0,08 %	Add	• 0 ms • 20 calls	• SunnyCalc.Maths.MathsService.Add(Double, Double)
0,05 %	Power	• 0 ms • 11 calls	• SunnyCalc.Maths.MathsService.Power(Double, UInt32)
0,02 %	Divide	• 0 ms • 2 calls	• SunnyCalc.Maths.MathsService.Divide(Double, Double)
0,01 %	Subtract	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Subtract(Int32, Int32)
0,01 %	Subtract	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Subtract(Double, Double)
0,01 %	Multiply	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Multiply(Double, Double)

Overview		Profiler.CalculateDoubles X	
▲ 100,00 %	CalculateDoubles	• 1 ms • 1 call	• SunnyCalc.Profiling.Profiler.CalculateDoubles
▼ 11,10 %	MoveNext	• 0 ms • 101 calls	• System.Collections.Generic.List+Enumerator`1.MoveNext
1,48 %	Root	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Root(Double, UInt32)
0,71 %	Add	• 0 ms • 200 calls	• SunnyCalc.Maths.MathsService.Add(Double, Double)
0,41 %	Power	• 0 ms • 101 calls	• SunnyCalc.Maths.MathsService.Power(Double, UInt32)
0,02 %	Divide	• 0 ms • 2 calls	• SunnyCalc.Maths.MathsService.Divide(Double, Double)
0,01 %	Subtract	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Subtract(Int32, Int32)
0,01 %	Multiply	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Multiply(Double, Double)
0,00 %	Subtract	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Subtract(Double, Double)

Overview		Profiler.CalculateDoubles X	
▲ 100,00 %	CalculateDoubles	• 1 ms • 1 call	• SunnyCalc.Profiling.Profiler.CalculateDoubles
▼ 10,99 %	MoveNext	• 0 ms • 1 001 calls	• System.Collections.Generic.List+Enumerator`1.MoveNext
5,29 %	Add	• 0 ms • 2 000 calls	• SunnyCalc.Maths.MathsService.Add(Double, Double)
2,62 %	Power	• 0 ms • 1 001 calls	• SunnyCalc.Maths.MathsService.Power(Double, UInt32)
1,24 %	Root	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Root(Double, UInt32)
0,01 %	Subtract	• 0 ms • 1 call	• SunnyCalc.Maths.MathsService.Subtract(Int32, Int32)
0,01 %	Divide	• 0 ms • 2 calls	• SunnyCalc.Maths.MathsService.Divide(Double, Double)
0,00 %	2 functions hidden • 0 ms total • 2 calls total		

Následující obsah této zprávy pojednává o profilování s velmi vysokým počtem vstupů (asi 14 000 000 čísel).

Samotný výpočet odchylky zabere při spuštění 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).

Visual Studio Performance Profiler interface showing the 'CalculateDoubles' method profile.

**Performance Profiler**

Snapshot | Process Overview | Top Methods | Show Threads | Show Events

Subsystems | 4 109 ms | Events | Thread State | Threads | CalculateDoubles | CPU 8,3%

0 | 1 s | 2 s | 3 s

**Methods and Subsystems under 'CalculateDoubles'**

Own / Total time

Search methods

74,6 %	Power • 801 ms • SunnyCalc.Maths.MathsService.Power(Double, UInt32)
12,2 %	CalculateDoubles • 131 ms / 1 074 ms • SunnyCalc.Profiling.Profiler.CalculateDoubles()
10,3 %	Add • 110 ms • SunnyCalc.Maths.MathsService.Add(Double, Double)
2,1 %	MoveNext • 22 ms • System.Collections.Generic.List`1+Enumerator.MoveNext()
0,9 %	get_Current • 9,8 ms • System.Collections.Generic.List`1+Enumerator.get_Current()
<0,01 %	GetEnumerator • 0,08 ms • System.Collections.Generic.List`1.GetEnumerator()

**User code**

**Call Tree**

Backtraces

100 %	CalculateDoubles • 1 074 ms • SunnyCalc.Profiling.Profiler.CalculateDoubles()
74,6 %	Power • 801 ms • SunnyCalc.Maths.MathsService.Power(Double, UInt32)
10,3 %	Add • 110 ms • SunnyCalc.Maths.MathsService.Add(Double, Double)
2,06 %	MoveNext • 22 ms • System.Collections.Generic.List`1+Enumerator.MoveNext()
1,31 %	[Unknown] • 14 ms
0,91 %	get_Current • 9,8 ms • System.Collections.Generic.List`1+Enumerator.get_Current()
0,23 %	coreclr.dll • 2,4 ms
0,01 %	GetEnumerator • 0,08 ms • System.Collections.Generic.List`1.GetEnumerator()
0,00 %	CalculateDoubles • 0,05 ms • SunnyCalc.Profiling.Profiler.CalculateDoubles()

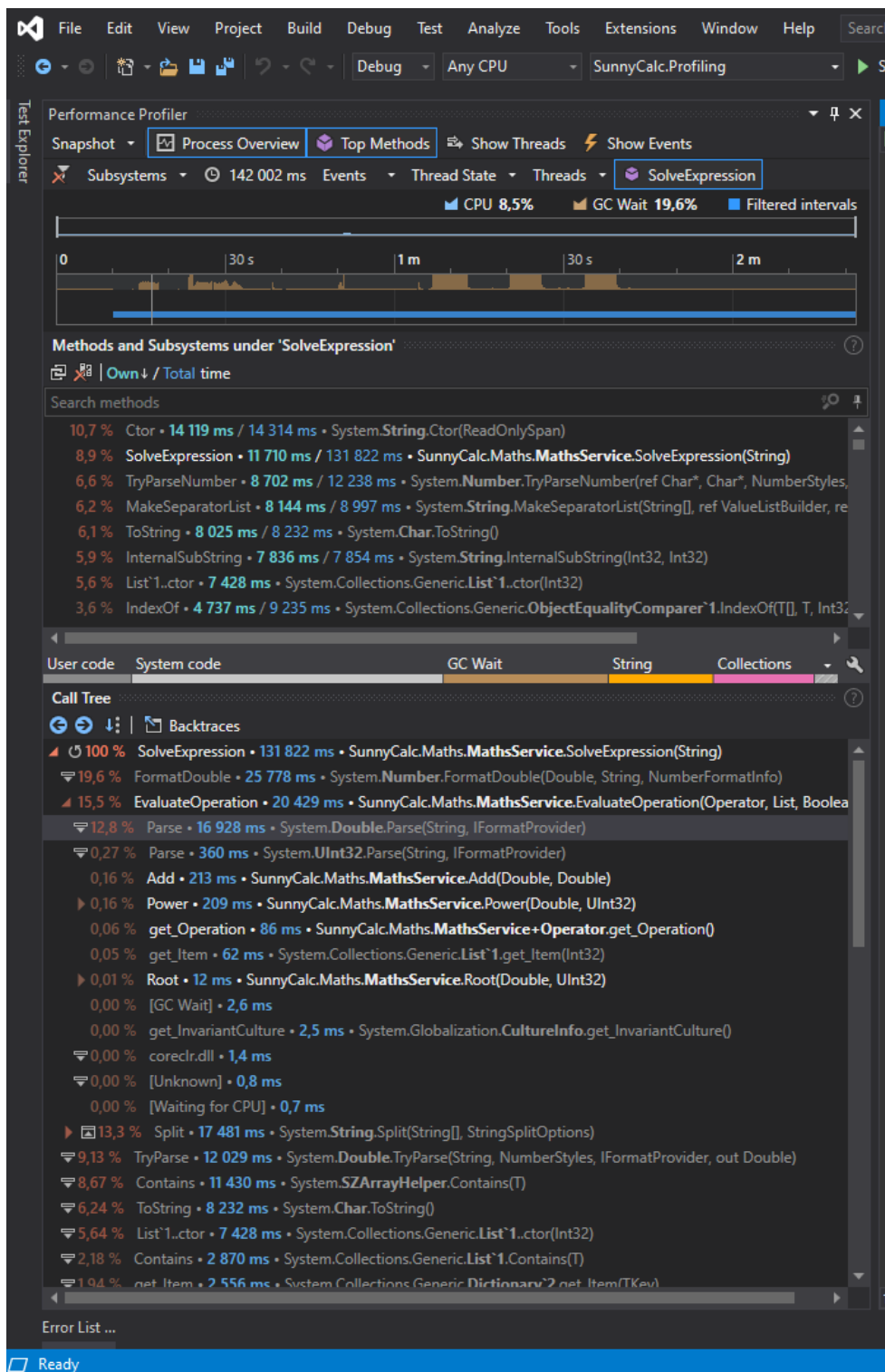
Error List ...

Item(s) Saved

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.



Visual Studio Performance Profiler interface showing the 'SolveExpression' method profile.

**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**

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