Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Асинхронное программирование
тема

Вариант 11

Преподаватель — подпись, дата — А. А. Чикизов (инициалы, фамилия)

Студент КИ18-16б 031831229 (номер группы, зачетной книжки) — подпись, дата — В. А. Прекель (инициалы, фамилия)

Красноярск 2020

1 Задание

Распараллелить выполнение программы из лабораторной работы 1.

2 Ход работы

Используется язык F# и готовые реализации алгоритма из первой лабораторной работы.

Вычисление 10000 значений от 1.0 до 1.000001 с шагом 0.0000000001 используя алгоритм, использующий векторные AVX-инструкции параллельно со степенью параллелизма 12 (равной числу логических ядер). Вывод времени выполнения в миллисекундах и первых 10 результатов.

Листинг 1 – CSharpLabs.Lab06/CSharpLabs.Lab06.FSharp/Program.fs

```
open System
open System.Diagnostics
open CSharpLabs.Lab01.Core.InverseHyperbolicCotangent

let calcAsync (x, eps) =
    async {
        let calc = ArcothAvx()
        let a = calc.Calculate(x, eps, Int32.MaxValue)
        return (x, a, calc.N, calc.Status)
    }

[<EntryPoint>]
let main argv =
    let sw = Stopwatch()
    sw.Start()

    let a =
        [ 1.0 .. 0.0000000001 .. 1.000001 ]
        |> Seq.map (fun i -> (i, 1e-9))
        |> Seq.map calcAsync
        |> (fun r -> Async.Parallel(r, 12))
        |> Async.RunSynchronously

    printfn "%f" sw.Elapsed.TotalMilliseconds

    printfn
        "%s"
        (a
         |> Array.take 10
         |> Array.map (fun (x, res, n, status) ->
             $" {x, 20} | {ArcothAvx.ReferenceFunction(x), 20} | {res, 20} | {n,
10} | {status, 20}")
         |> String.concat "\n")

    0
```

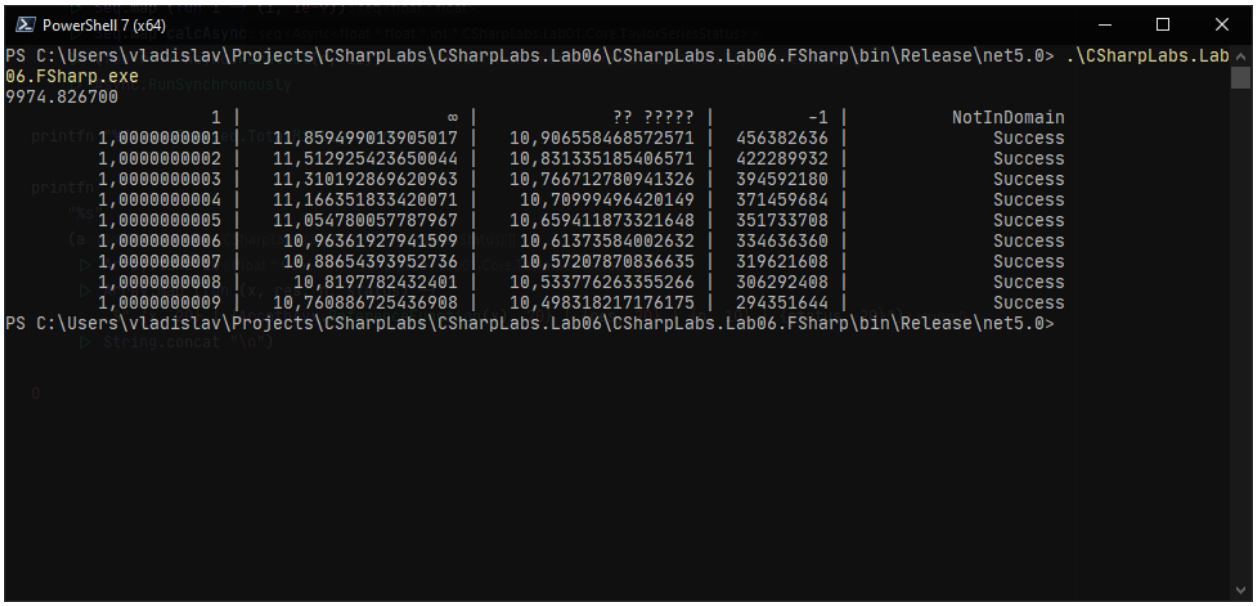

Рисунок 1 – Выполнение со степенью параллелизма 12

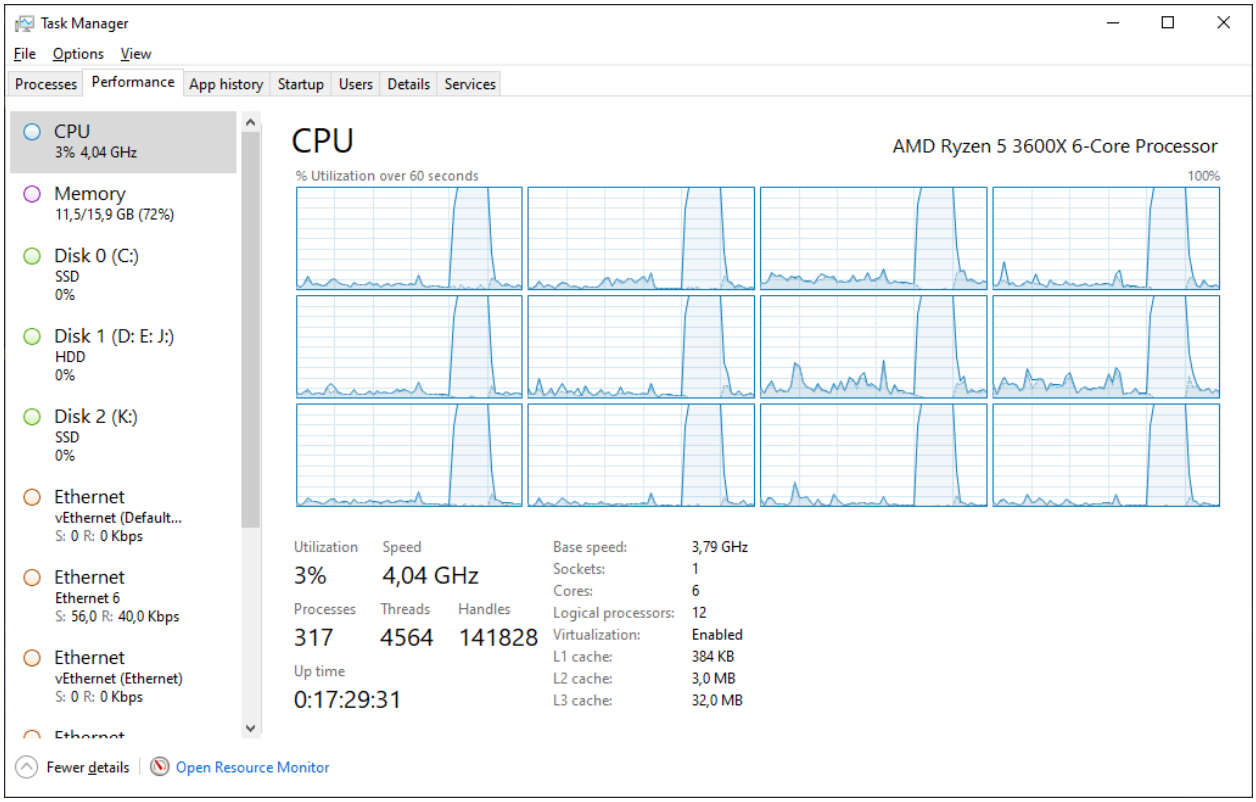

Рисунок 2 – Загрузка процессора со степенью параллелизма 12

В строчке (fun r -> Async.Parallel(r, 12)) число - это степень параллелизма. Запустим со степенью 6 и 1.

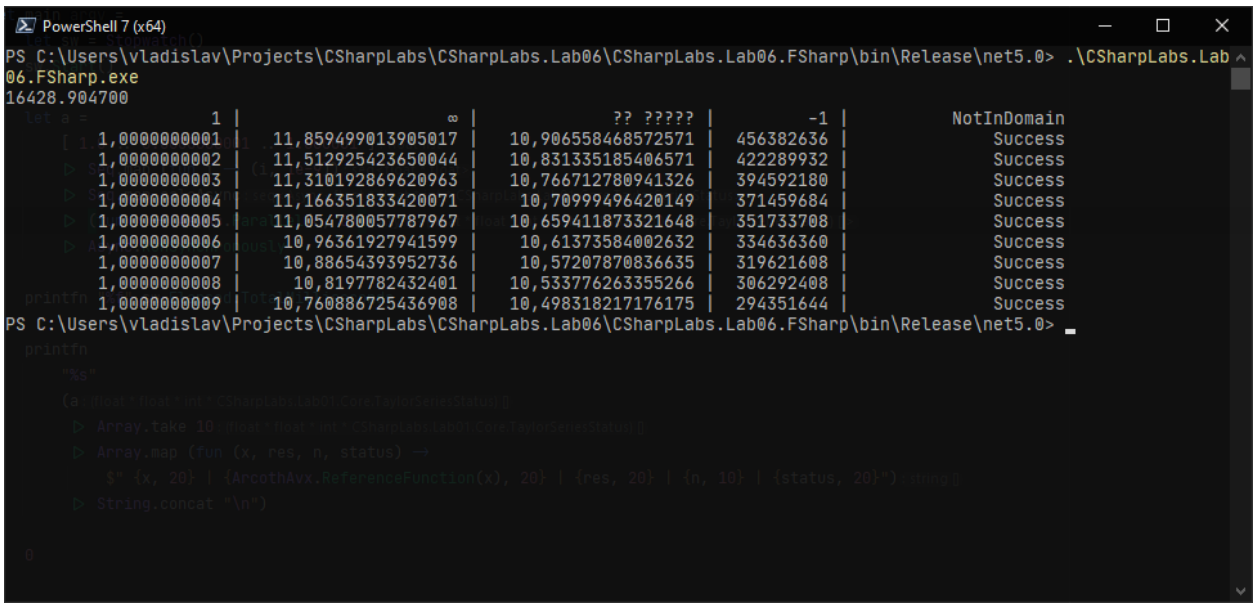

Рисунок 3 – Выполнение со степенью параллелизма 6

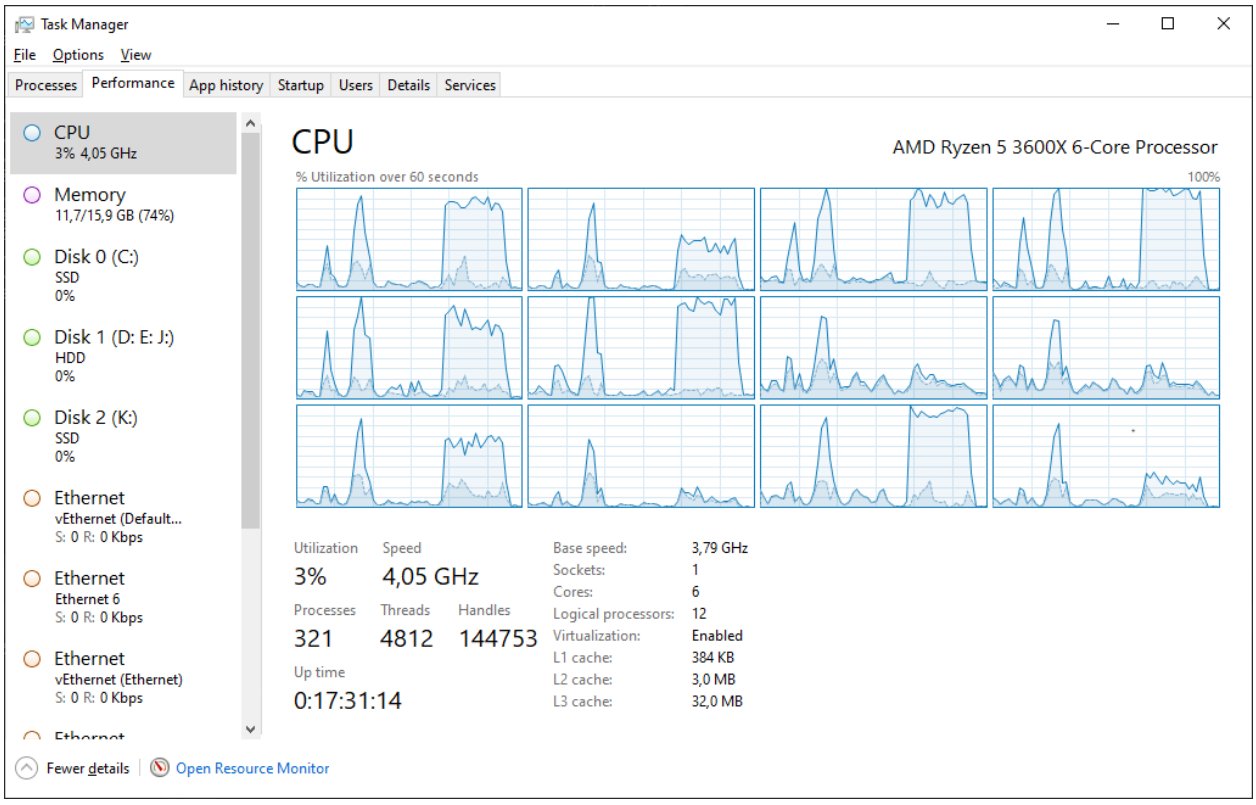

Рисунок 4 – Загрузка процессора со степенью параллелизма 6

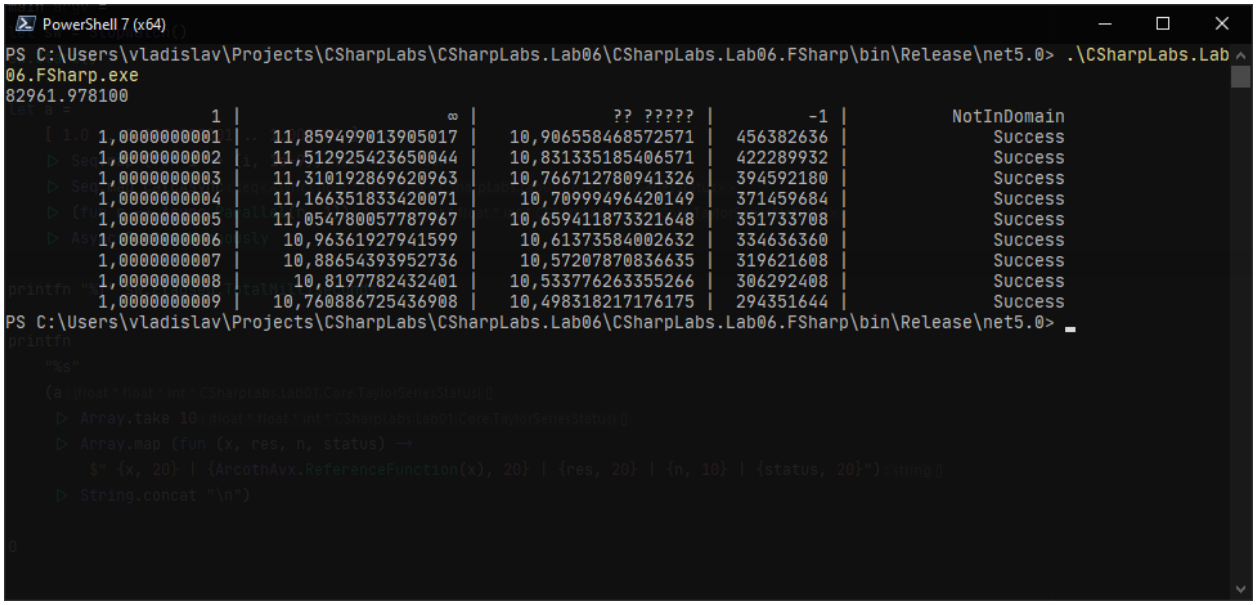

Рисунок 5 – Выполнение со степенью параллелизма 1

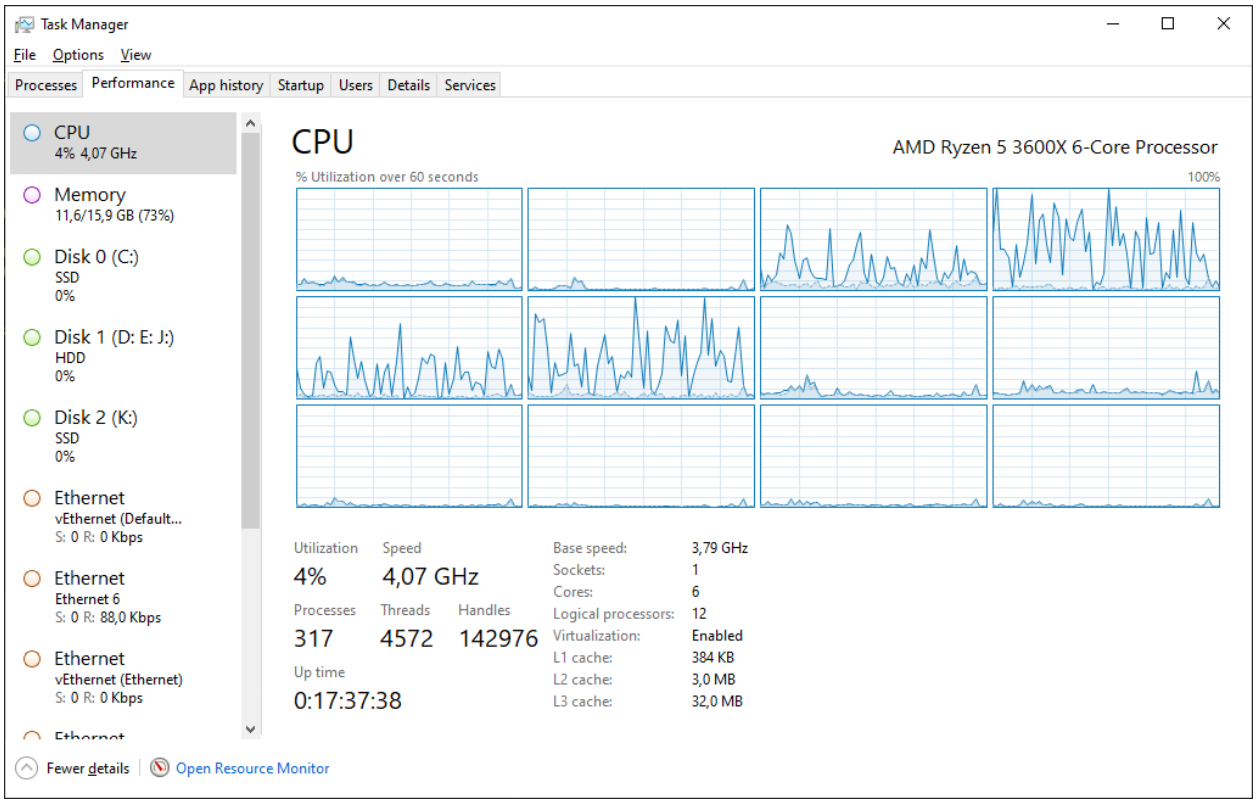

Рисунок 6 – Загрузка процессора со степенью параллелизма 1

Так же написан бенчмарк на фреймворке BenchmarkDotNet. В измеряемом методе вычисляется для 120 значений от 1.0000000 до 1.0000120 с 0.0000001 двумя способами: с использованием AVX инструкций и без. Измеряется для разной степени парралелизма от 1 до 24.

Листинг 2 – CSharpLabs.Lab06/CSharpLabs.Lab06.FSharp.Benchmark/ArcothArcothBenchmark.fs

```
module ArcothArcothBenchmark

open System
open BenchmarkDotNet.Attributes

open CSharpLabs.Lab01.Core.InverseHyperbolicCotangent

let calcAsync (calcCreator: (unit -> AbstractArcoth)) x =
    async {
        let calc = calcCreator ()
        calc.Calculate(x, 1e-7, Int32.MaxValue) |> ignore
    }

[<HtmlExporter>]
[<CsvExporter>]
[<RPlotExporter>]
type ArcothArcothBenchmark() =
    let arr =
        [| 1.0000000 .. 0.0000001 .. 1.0000120 |]

    member val public CalcCreators: (unit -> AbstractArcoth) list = [ fun () ->
upcast (ArcothAvx())
                                                                      //fun () -
> upcast (ArcothLinq())
                                                                      //fun () -
> upcast (ArcothNaive())
                                                                      fun () ->
upcast (ArcothOptimized()) ] with get, set

    [<ParamsSource("CalcCreators")>]
    member val public CalcCreator: (unit -> AbstractArcoth) = fun () -> upcast
(ArcothAvx()) with get, set

    member val public Degrees = [ 1 .. Environment.ProcessorCount * 2 ] with
get, set

    [<ParamsSource("Degrees")>]
    member val public Degree = 0 with get, set

    [<Benchmark>]
    member this.BenchmarkParallelDegree() =
```

```
        arr
        |> Seq.map (calcAsync this.CalcCreator)
        |> (fun r -> Async.Parallel(r, this.Degree))
        |> Async.RunSynchronously
```

Результаты:

```
BenchmarkDotNet=v0.12.1, OS=Windows 10.0.19042
AMD Ryzen 5 3600X, 1 CPU, 12 logical and 6 physical cores
.NET Core SDK=5.0.200-preview.20601.7
  [Host]     : .NET Core 5.0.0 (CoreCLR 5.0.20.51904, CoreFX 5.0.20.51904), X64
RyuJIT DEBUG
  DefaultJob : .NET Core 5.0.0 (CoreCLR 5.0.20.51904, CoreFX 5.0.20.51904), X64
RyuJIT
```

Method	CalcCreator	Degree	Mean	Error	StdDev
BenchmarkParallelDegree	ArcothAvx	1	41.150 ms	0.0300 ms	0.0281 ms
BenchmarkParallelDegree	ArcothAvx	2	20.907 ms	0.0112 ms	0.0093 ms
BenchmarkParallelDegree	ArcothAvx	3	14.314 ms	0.0487 ms	0.0456 ms
BenchmarkParallelDegree	ArcothAvx	4	10.911 ms	0.0631 ms	0.0590 ms
BenchmarkParallelDegree	ArcothAvx	5	8.959 ms	0.1215 ms	0.1015 ms
BenchmarkParallelDegree	ArcothAvx	6	8.005 ms	0.0428 ms	0.0380 ms
BenchmarkParallelDegree	ArcothAvx	7	7.205 ms	0.0257 ms	0.0240 ms
BenchmarkParallelDegree	ArcothAvx	8	6.550 ms	0.0102 ms	0.0085 ms
BenchmarkParallelDegree	ArcothAvx	9	6.057 ms	0.0207 ms	0.0183 ms
BenchmarkParallelDegree	ArcothAvx	10	5.629 ms	0.0094 ms	0.0088 ms
BenchmarkParallelDegree	ArcothAvx	11	5.325 ms	0.0041 ms	0.0036 ms
BenchmarkParallelDegree	ArcothAvx	12	5.051 ms	0.0256 ms	0.0240 ms
BenchmarkParallelDegree	ArcothAvx	13	5.051 ms	0.0198 ms	0.0166 ms
BenchmarkParallelDegree	ArcothAvx	14	5.067 ms	0.0194 ms	0.0162 ms
BenchmarkParallelDegree	ArcothAvx	15	5.077 ms	0.0437 ms	0.0388 ms
BenchmarkParallelDegree	ArcothAvx	16	5.070 ms	0.0365 ms	0.0323 ms
BenchmarkParallelDegree	ArcothAvx	17	5.069 ms	0.0539 ms	0.0450 ms
BenchmarkParallelDegree	ArcothAvx	18	5.075 ms	0.0322 ms	0.0301 ms
BenchmarkParallelDegree	ArcothAvx	19	5.103 ms	0.0253 ms	0.0198 ms
BenchmarkParallelDegree	ArcothAvx	20	5.122 ms	0.0648 ms	0.0606 ms
BenchmarkParallelDegree	ArcothAvx	21	5.154 ms	0.0697 ms	0.0652 ms
BenchmarkParallelDegree	ArcothAvx	22	5.101 ms	0.0369 ms	0.0345 ms
BenchmarkParallelDegree	ArcothAvx	23	5.105 ms	0.0561 ms	0.0524 ms
BenchmarkParallelDegree	ArcothAvx	24	5.073 ms	0.0284 ms	0.0266 ms
BenchmarkParallelDegree	ArcothOptimized	1	165.103 ms	0.7043 ms	0.6588 ms
BenchmarkParallelDegree	ArcothOptimized	2	83.361 ms	0.1112 ms	0.1040 ms
BenchmarkParallelDegree	ArcothOptimized	3	56.651 ms	0.2808 ms	0.2627 ms
BenchmarkParallelDegree	ArcothOptimized	4	43.085 ms	0.2665 ms	0.2493 ms
BenchmarkParallelDegree	ArcothOptimized	5	35.449 ms	0.4704 ms	0.4400 ms
BenchmarkParallelDegree	ArcothOptimized	6	31.246 ms	0.2655 ms	0.2483 ms
BenchmarkParallelDegree	ArcothOptimized	7	28.218 ms	0.1016 ms	0.0901 ms

Method	CalcCreator	Degree	Mean	Error	StdDev
BenchmarkParallelDegree	ArcothOptimized	8	25.427 ms	0.4925 ms	0.4837 ms
BenchmarkParallelDegree	ArcothOptimized	9	23.584 ms	0.3162 ms	0.2803 ms
BenchmarkParallelDegree	ArcothOptimized	10	21.712 ms	0.0287 ms	0.0240 ms
BenchmarkParallelDegree	ArcothOptimized	11	20.590 ms	0.0864 ms	0.0674 ms
BenchmarkParallelDegree	ArcothOptimized	12	19.818 ms	0.1082 ms	0.0959 ms
BenchmarkParallelDegree	ArcothOptimized	13	19.818 ms	0.1235 ms	0.1095 ms
BenchmarkParallelDegree	ArcothOptimized	14	19.814 ms	0.1421 ms	0.1329 ms
BenchmarkParallelDegree	ArcothOptimized	15	19.750 ms	0.0703 ms	0.0658 ms
BenchmarkParallelDegree	ArcothOptimized	16	19.728 ms	0.1900 ms	0.1587 ms
BenchmarkParallelDegree	ArcothOptimized	17	19.631 ms	0.0777 ms	0.0688 ms
BenchmarkParallelDegree	ArcothOptimized	18	19.800 ms	0.1381 ms	0.1292 ms
BenchmarkParallelDegree	ArcothOptimized	19	19.809 ms	0.0717 ms	0.0671 ms
BenchmarkParallelDegree	ArcothOptimized	20	19.778 ms	0.0687 ms	0.0609 ms
BenchmarkParallelDegree	ArcothOptimized	21	19.744 ms	0.0787 ms	0.0736 ms
BenchmarkParallelDegree	ArcothOptimized	22	19.819 ms	0.1859 ms	0.1648 ms
BenchmarkParallelDegree	ArcothOptimized	23	19.630 ms	0.1138 ms	0.0889 ms
BenchmarkParallelDegree	ArcothOptimized	24	19.858 ms	0.1084 ms	0.0961 ms

Вывод:

Использование параллелизма увеличивает скорость выполнения на многоядерных системах.