

Simple example of serial and parallel sorting of integer arrays

Sorting for variety of Merge Sort(s) results are equal

Examples of sorting of user defined classes

Unsorted array of user defined class: (16 : 0)(12 : 1)(18 : 2)(18 : 3)(10 : 4)( 2 : 5)

Sorted array of user defined class: ( 2 : 5)(10 : 4)(12 : 1)(16 : 0)(18 : 2)(18 : 3)

Unsorted array of user defined class: (16 : 0)(12 : 1)(18 : 2)(18 : 3)(10 : 4)( 2 : 5)

Sorted array of user defined class: ( 2 : 5)(10 : 4)(12 : 1)(16 : 0)(18 : 2)(18 : 3)

Performance comparison of Merge Sort

C# array of size 10000000: Array.Sort 0.928 sec, Serial Merge Sort 1.873 sec, speedup 0.50

C# array of size 10000000: Array.Sort 0.929 sec, Parallel Merge Sort 0.173 sec, speedup 5.35

C# array of size 10000000: Linq.OrderBy 4.441 sec, Serial Merge Sort 1.814 sec, speedup 2.45

C# array of size 10000000: Linq.OrderBy.AsParallel 1.375 sec, Parallel Merge Sort 0.157 sec, speedup 8.76

Performance comparison of Serial and Parallel Radix Sort for Array

C# array of size 10000000: Array.Sort 0.925 sec, Serial Radix Sort 0.121 sec, speedup 7.67

C# array of size 10000000: Linq.OrderBy 4.530 sec, Serial Radix Sort 0.119 sec, speedup 38.10

C# array of size 10000000: Linq.OrderBy.AsParallel 1.373 sec, Parallel Radix Sort 0.066 sec, speedup 20.66

Performance comparison of Serial Radix Sort for List

C# List of size 10000000: List.Sort 0.918 sec, Serial Radix Sort 0.156 sec, speedup 5.89

Performance comparison of Merge Sort of User Defined Class

C# array of size 1000000: Array.Sort 0.305 sec, Serial Merge Sort 0.338 sec, speedup 0.90

C# array of size 1000000: Array.Sort 0.321 sec, Parallel Merge Sort 0.067 sec, speedup 4.79

C# array of size 1000000: Linq.OrderBy 0.386 sec, Serial Merge Sort 0.339 sec, speedup 1.14

C# array of size 1000000: Linq.OrderBy.AsParallel 0.404 sec, Parallel Merge Sort 0.131 sec, speedup 3.07

Performance comparison of Serial Radix Sort for Array of User Defined Class

C# array of size 1000000: Array.Sort 0.303 sec, Serial Radix Sort 0.100 sec, speedup 3.04

C# array of size 1000000: Linq.OrderBy 0.352 sec, Serial Radix Sort 0.094 sec, speedup 3.73

C# array of size 1000000: Linq.OrderBy.AsParallel 0.339 sec, Serial Radix Sort 0.096 sec, speedup 3.55

Example of sequence equality checking - serial and parallel

Check array equality: True True

Check array inequality: False False

Check list equality: True True

Check list inequality: False False

Example of sequence Min/Max - serial and parallel

Check array Min: -3 -3

Check List Min: -3 -3

Check array Max: 43 43

Check List Max: 43 43

Performance comparison of sequence equality and Min/Max - serial and parallel

C# array of size 16777216: Linq.Equal 0.201 sec, HPC.Equal 0.036 sec, speedup 5.53

C# List of size 16777216: Linq.Equal 0.287 sec, HPC.Equal 0.059 sec, speedup 4.90

C# array of size 16777216: Linq.Min 0.091 sec, HPC.Min 0.044 sec, speedup 2.06

C# List of size 16777216: Linq.Min 0.130 sec, HPC.Min 0.051 sec, speedup 2.53

C# array of size 16777216: Linq.Max 0.078 sec, HPC.Max 0.041 sec, speedup 1.92

C# List of size 16777216: Linq.Max 0.122 sec, HPC.Max 0.046 sec, speedup 2.63

First loop sum = 31

First .Sum() = 31

Sum Naive (for loop) = 0, correct answer is 2.0

Sum Kahan = 2, correct answer is 2.0

Sum Kahan (SSE) = 2, correct answer is 2.0

Sum Kahan (SSE & multi-core) = 2, correct answer is 2.0

C# array of size 67108864: Sum 0.334 sec, HPC#.SumToLong 0.039 sec, speedup 8.51

C# array of size 67108864: Sum 0.334 sec, HPC#.SumToLongSse 0.020 sec, speedup 17.04

C# array of size 67108864: Sum.AsParallel 0.152 sec, HPC#.SumToLongSse 0.020 sec, speedup 7.75

C# array of size 67108864: Sum.AsParallel 0.152 sec, HPC#.SumToLongSsePar 0.012 sec, speedup 12.90