Odd Even Sorting Implementation

Marius Wichtner, Anton Beck, Akif Bagci,
Eren Albayrak

02.02.2018

FRANKFURT
UNIVERSITY
OF APPLIED SCIENCES

Overview

Introduction

Algorithm

Implementation

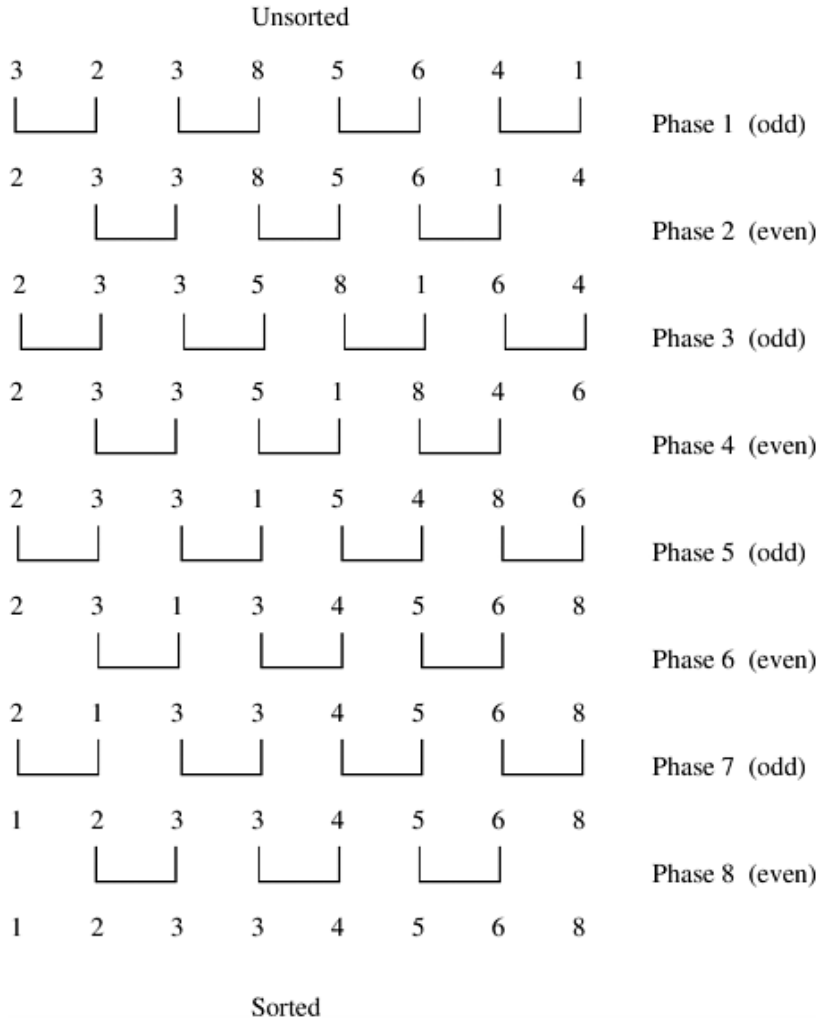
Scalability

Conclusion

Introduction

- Team
- Motivation
- Task

Algorithm - Not Parallel



Parallel Implementation

t	P0		P1		P2		P3		
0	7	6	3	9	1	8	4	2	
1	3	6	7	9	1	2	4	8	even
2	3	6	1	2	7	9	4	8	odd
3	1	2	3	6	4	7	8	9	even
4	1	2	3	4	6	7	8	9	odd

Profiling Python

5 cores -> 45.000.000 digits

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     10    5.896    0.590    5.896    0.590 {method 'sort' of 'numpy.ndarray' objects}
      5    2.585    0.517    2.585    0.517 {method 'Sendrecv' of 'mpi4py.MPI.Comm' objects}
     40    1.246    0.031    1.246    0.031 {method 'copy' of 'numpy.ndarray' objects}
      5    0.853    0.171    0.853    0.171 {built-in method numpy.core.multiarray.concatenate}
      1    0.753    0.753    0.753    0.753 {method 'Gather' of 'mpi4py.MPI.Comm' objects}
     11    0.666    0.061    0.671    0.061 {built-in method _imp.create_dynamic}
      1    0.535    0.535    0.535    0.535 {built-in method numpy.core.multiarray.fromfile}
      1    0.150    0.150   10.826   10.826 parallel-odd-even-sort.py:42(do_odd_even_sort)
```

15 cores -> 45.000.000 digits

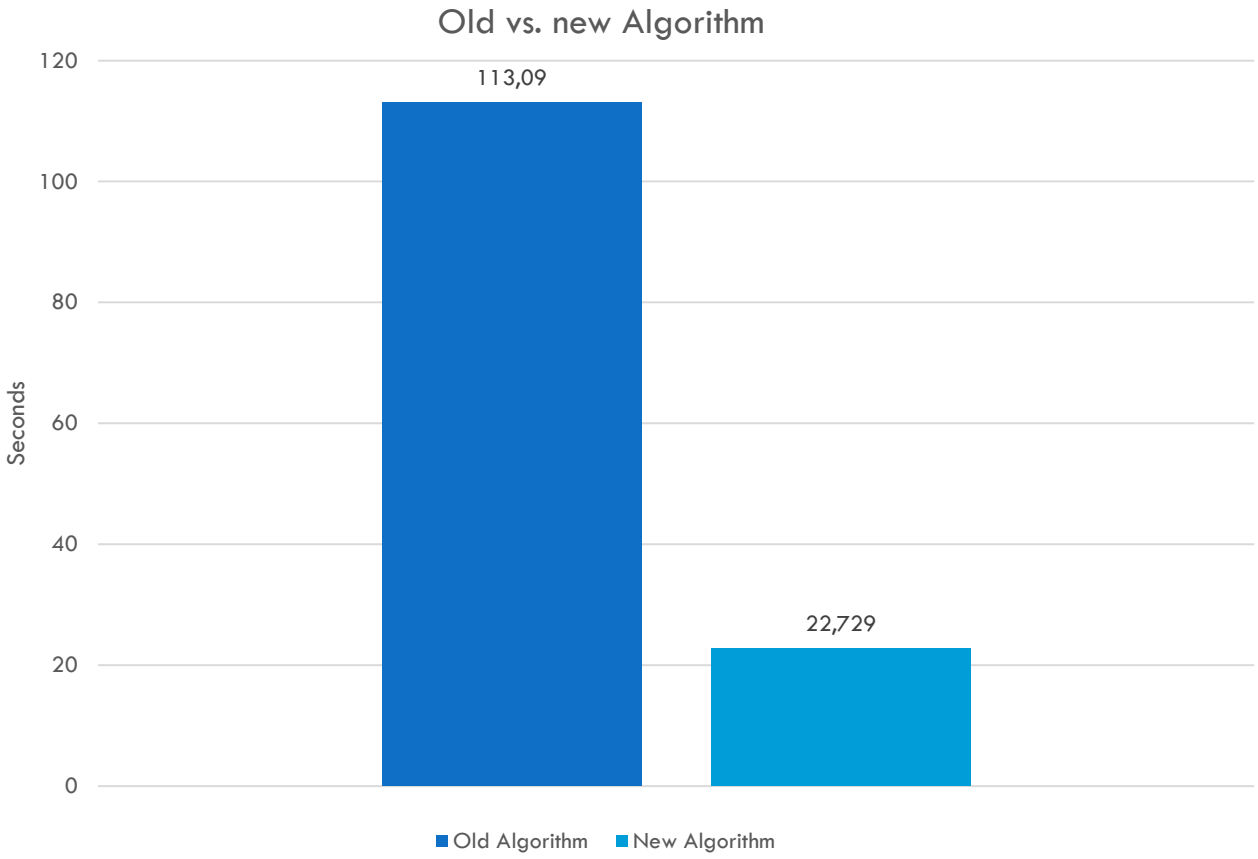
```
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     8    7.713    0.964    7.713    0.964 {method 'Sendrecv' of 'mpi4py.MPI.Comm' objects}
     8    3.393    0.424    3.393    0.424 {method 'sort' of 'numpy.ndarray' objects}
    11    1.952    0.177    1.957    0.178 {built-in method _imp.create_dynamic}
     1    1.503    1.503    1.503    1.503 {built-in method numpy.core.multiarray.fromfile}
     1    1.058    1.058    1.058    1.058 {method 'Gather' of 'mpi4py.MPI.Comm' objects}
    38    0.707    0.019    0.707    0.019 {method 'copy' of 'numpy.ndarray' objects}
    10    0.645    0.065    0.645    0.065 {built-in method numpy.core.multiarray.concatenate}
```

MPI with C++ & vector Advices

- Read complexity of every method you use
  - Insert/delete elements at position x: O(n)
    n = elements which have to move
  - Deleting triggers reduction of container size too
- Use: vector.at(i++) = value
  instead: vector.push_back(value)
  Because by larger sizes (>1gb): bad alloc exception
- Filling in arrays is faster than in vectors
- Memory allocation for big sizes: dynamic, not static

MPI with C++ & vector Advices

- Old Algorithm: 500 nodes and 5.119.997
- New Algorithm: 480 nodes and 4.800.000

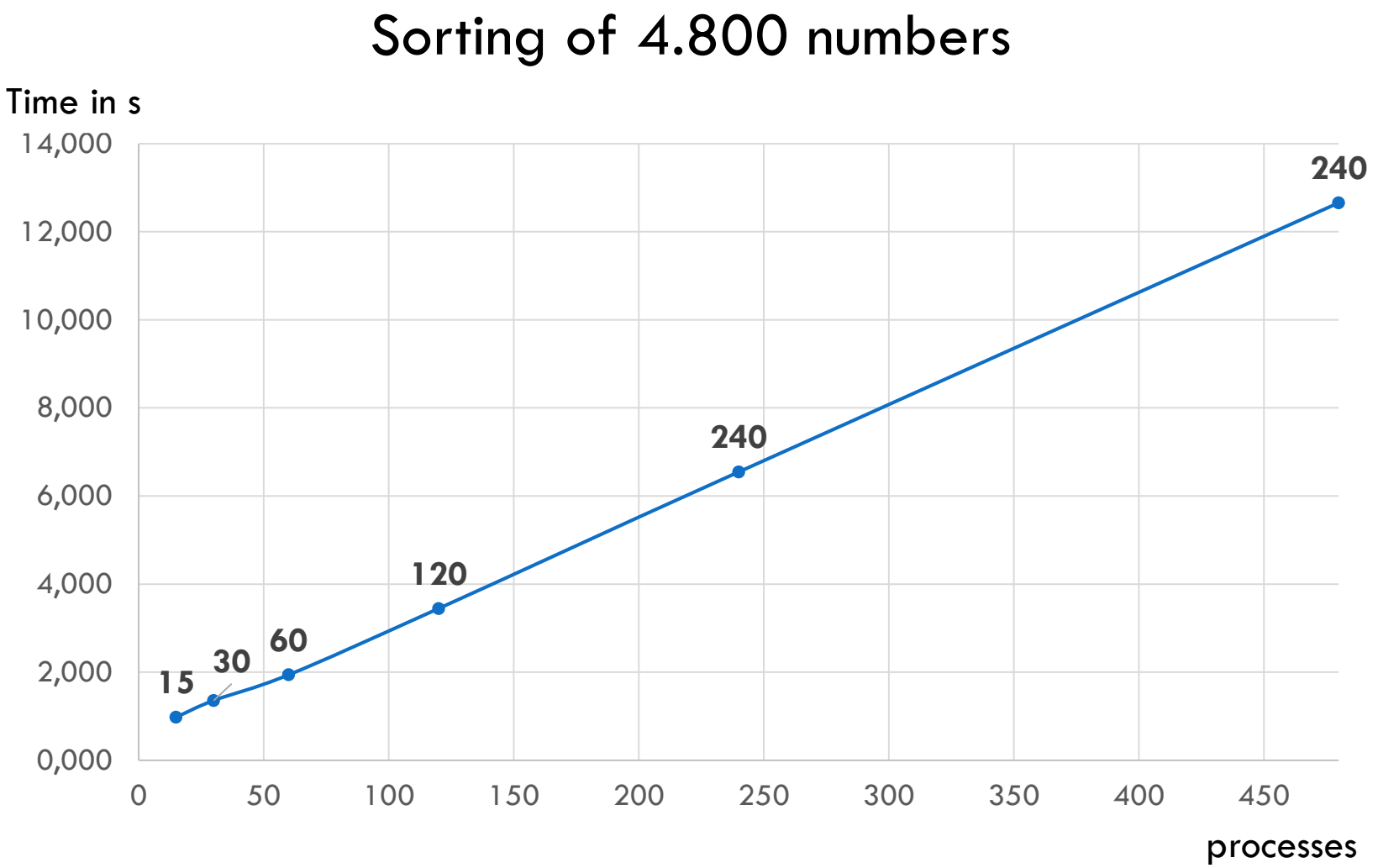


Scalability

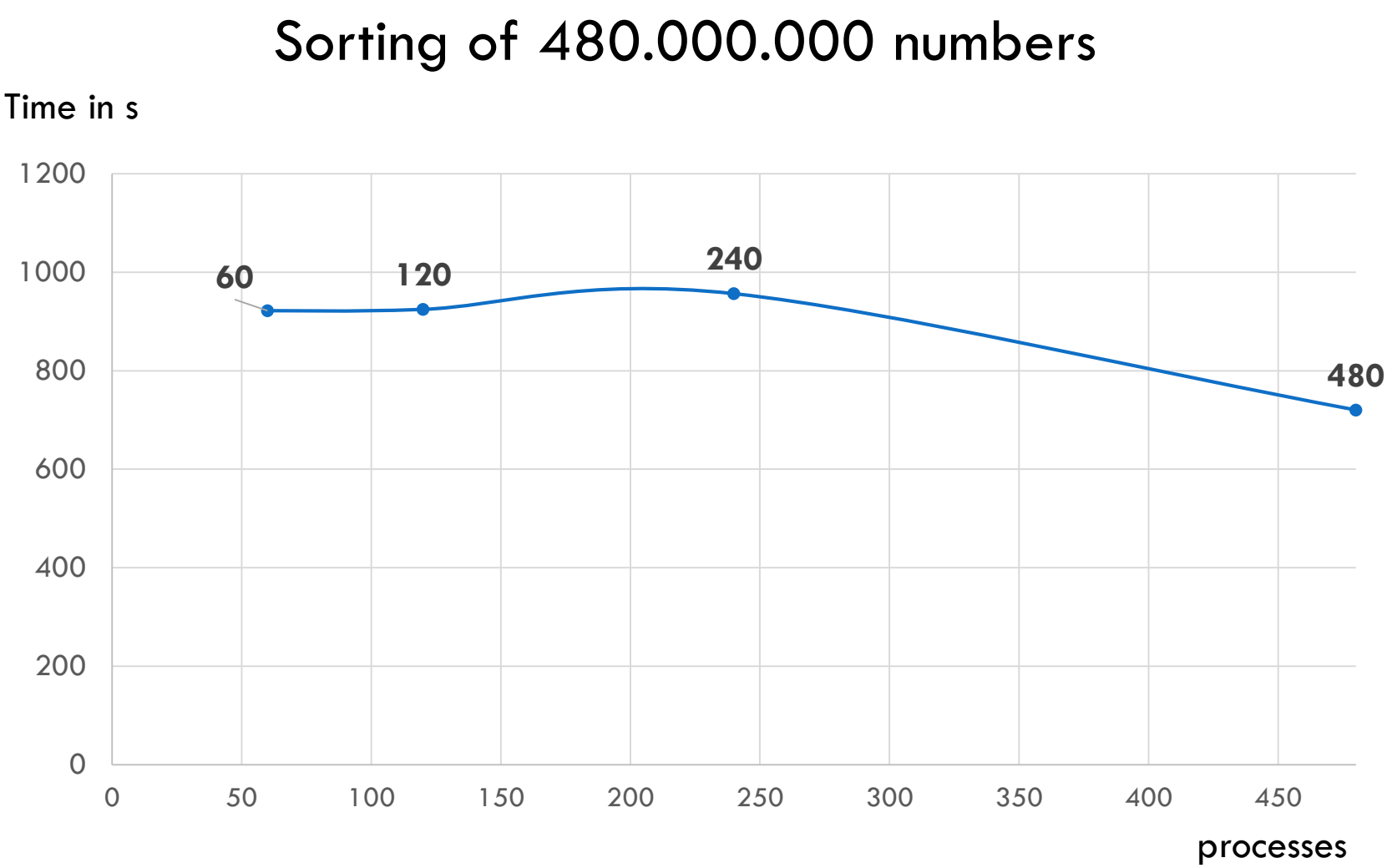
Required time for sorting in seconds

Count Processes:	15	30	60	120	240	480
4.800	0,976	1,355	1,938	3,444	6,545	12,657
48.000	1,027	1,368	2,064	3,539	6,775	12,711
480.000	1,630	2,001	2,662	4,071	7,441	13,651
4.800.000	7,566	7,840	8,545	11,537	15,479	22,729
48.000.000	71,458	71,589	71,076	75,547	103,359	105,701
480.000.000			922,072	924,685	956,681	720,331

Scalability



Scalability



Scalability

Speed: digits / nodes / seconds

I.e: For 4.800 digits with 15 nodes, each node sort in 1 sec. 327,869 digits

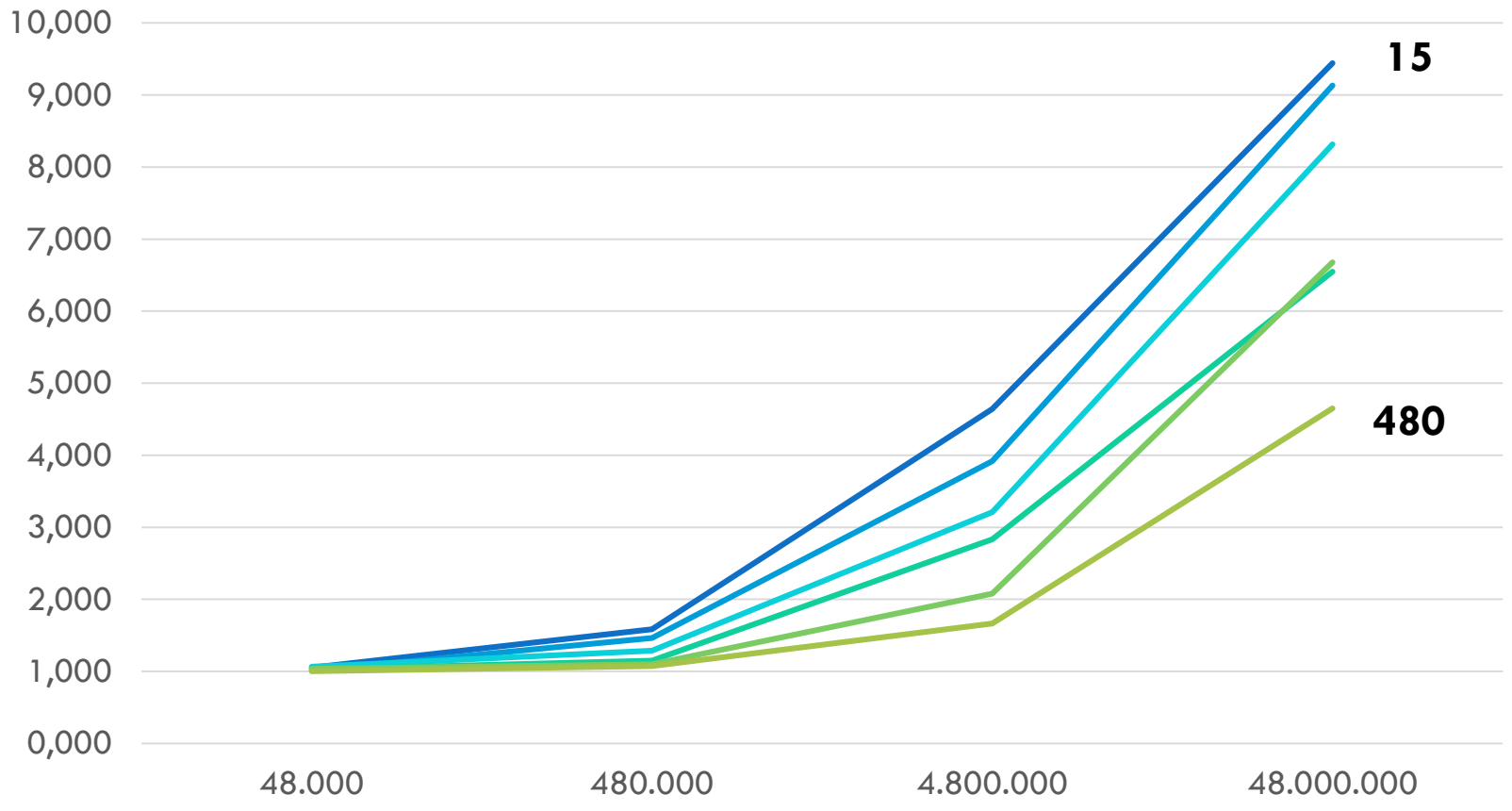
	15	30	60	120	240	480
4.800	327,869	118,081	41,280	11,614	3,056	0,790
48.000	3.115,871	1.169,591	387,597	113,026	29,520	7,867
480.000	19.631,902	7.996,002	3.005,259	982,560	268,781	73,255
4.800.000	42.294,475	20.408,163	9.362,200	3.467,106	1.292,073	439,967
48.000.000	44.781,550	22.349,802	11.255,557	5.294,717	1.935,003	946,065
480.000.000			8.676,112	4.325,797	2.090,561	1.388,251

Scalability

Scaling factors

	15	30	60	120	240	480
4.800	-	-	-	-	-	-
48.000	1,052	1,010	1,065	1,028	1,035	1,004
480.000	1,587	1,463	1,290	1,150	1,098	1,074
4.800.000	4,642	3,918	3,210	2,834	2,080	1,665
48.000.000	9,445	9,131	8,318	6,548	6,677	4,650
480.000.000	-	-	12,973	12,240	9,256	6,815

Scalability



Conclusion

Image Sources

- http://www.hedgeco.net/news/wp-content/uploads/2010/08/pic-page31.jpg
- https://www.geeksforgeeks.org/wp-content/uploads/Even-Odd-Sort.gif

Our Github Repo: https://github.com/erenalbayrak/Odd-Even-Sort-mit-MPI