MPI Special Challenge 2

Develop a parallel application that does find prime numbers by using the sieve of Eratosthenes with C and MPI

Participants: Khiem Truong Huu
Yannick Lamprecht
Christoph Schäfer
Florian Then
Harinath Vutla

Overview

- Sieve of Eratosthenes
- How to separate the work?
- Result of the Scaling analysis of the algorithm

Sieve of Eratosthenes

- K incremented in between of 2 and sqrt(number of given numbers)
- Repeat:
  - Mark all multiplies of K between 2*K and N (here red)
  - Set K to the smallest unmarked number
- All unmarked numbers are primes (here green)

K=2

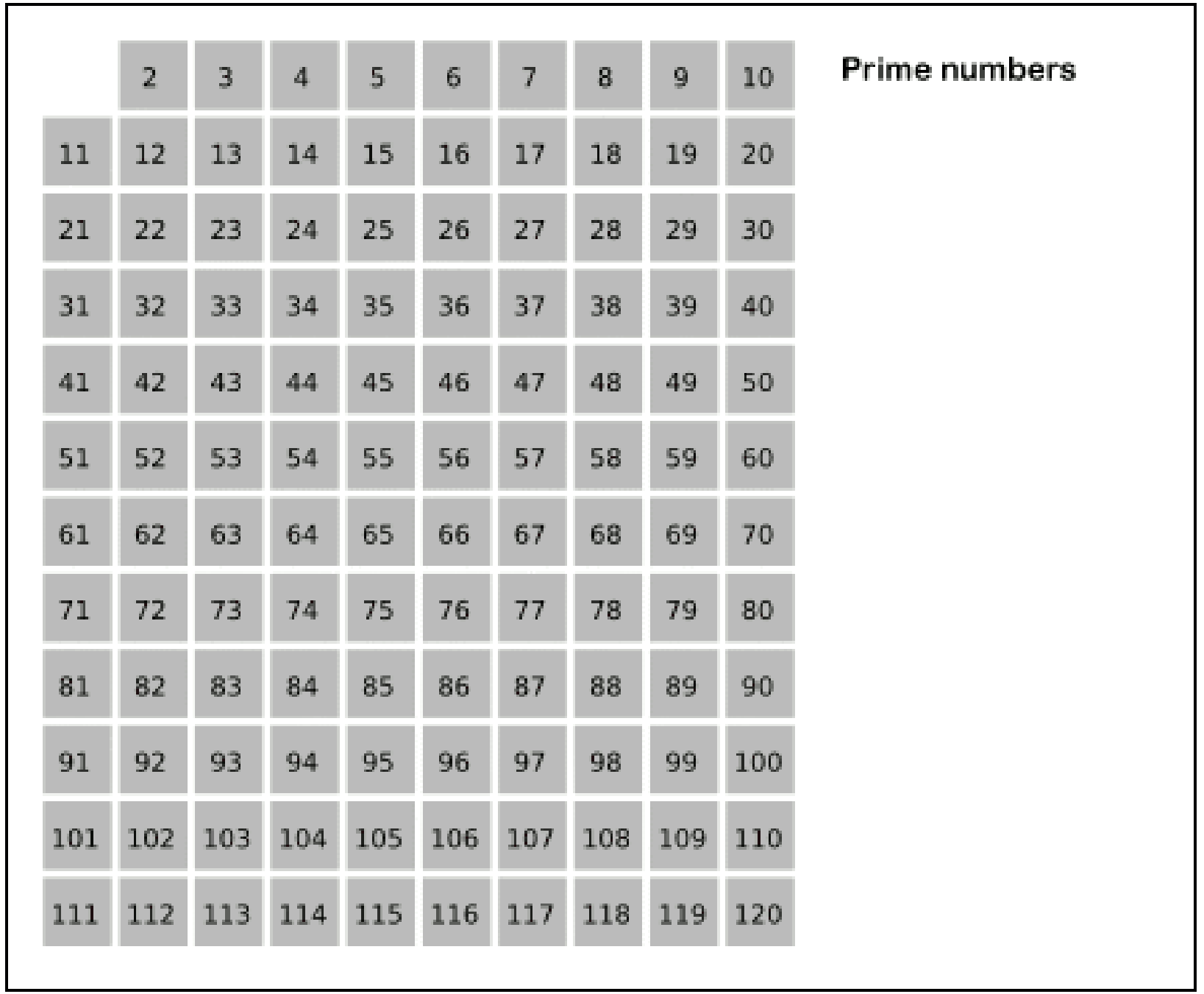
2	3	4
5	6	7
8	9	10
11	12	13

K=3

2	3	4
5	6	7
8	9	10
11	12	13

K>=sqrt(13)

2	3	4
5	6	7
8	9	10
11	12	13


2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109 110
111 112 113 114 115 116 117 118 119 120
Prime numbers

How to separate the work? Option 1

Split the tasks "round robin"

- Sqrt(n) tasks with p processes
  - Each process gets about (sqrt(n)/p) tasks to compete k
  - Leads to load imbalance
- With p = 4:
  - p0 has tasks with values 2, 6, 10, … // done after first step
  - p1 has tasks with values 3, 7, 11, …
  - p2 has values 4, 8, 12, ... //done after first step
  - p3 has values 5, 9, 13, …

How to separate the work? Option 2

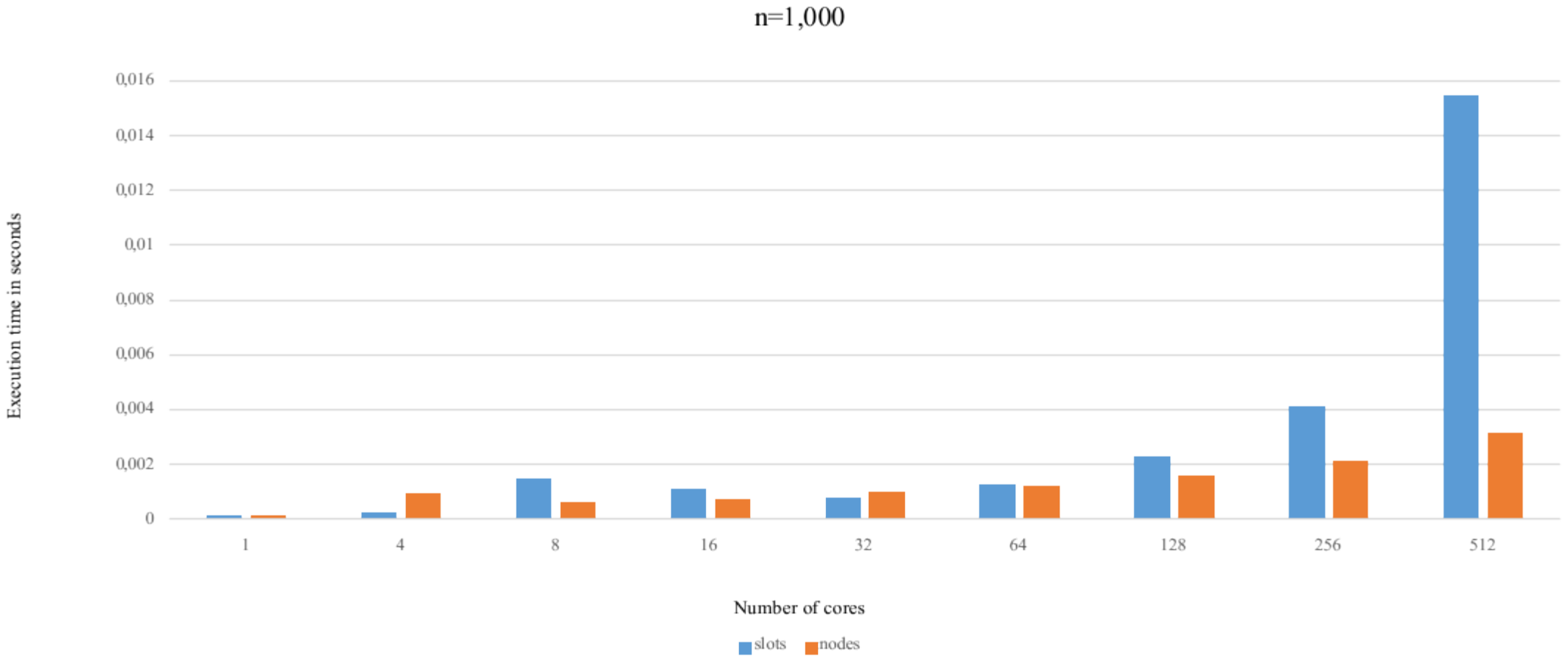
Split the input to blocks

17 elements divided among 7 processes

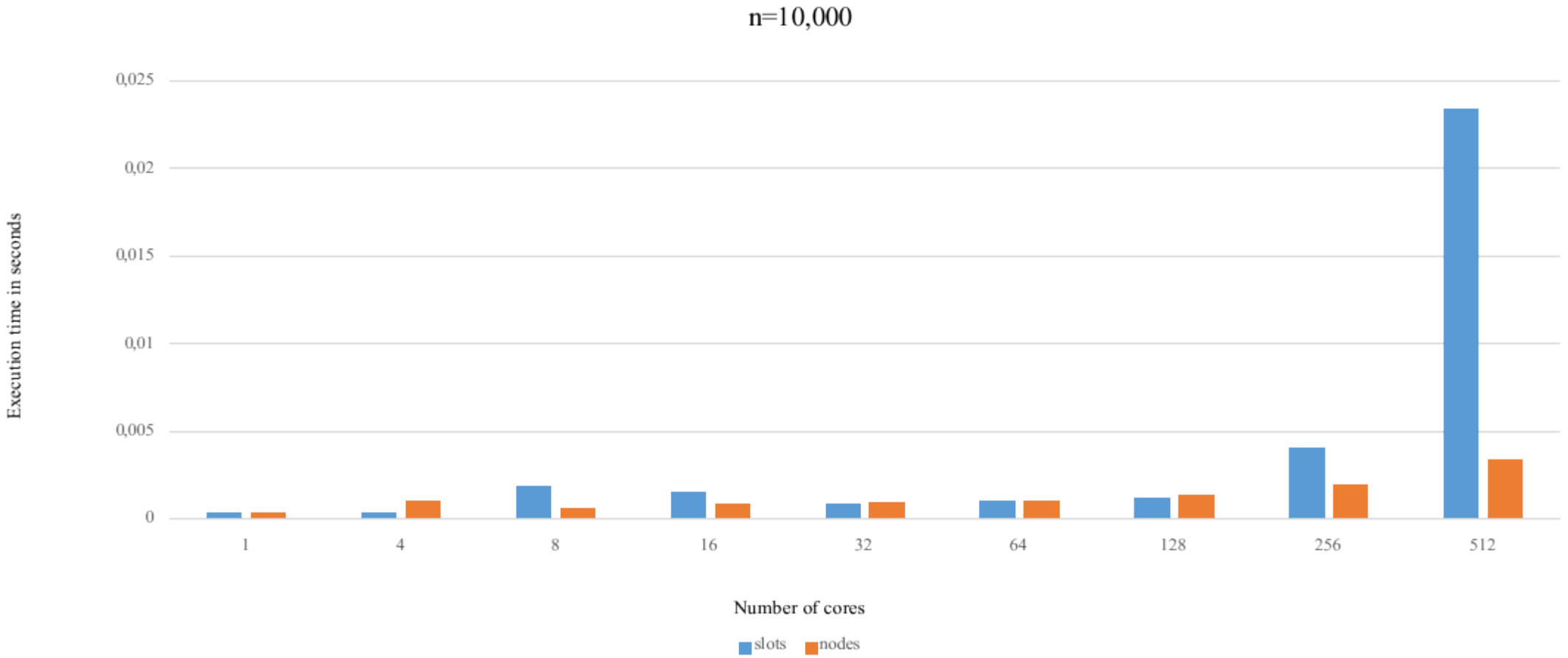
17 elements divided among 5 processes

17 elements divided among 3 processes

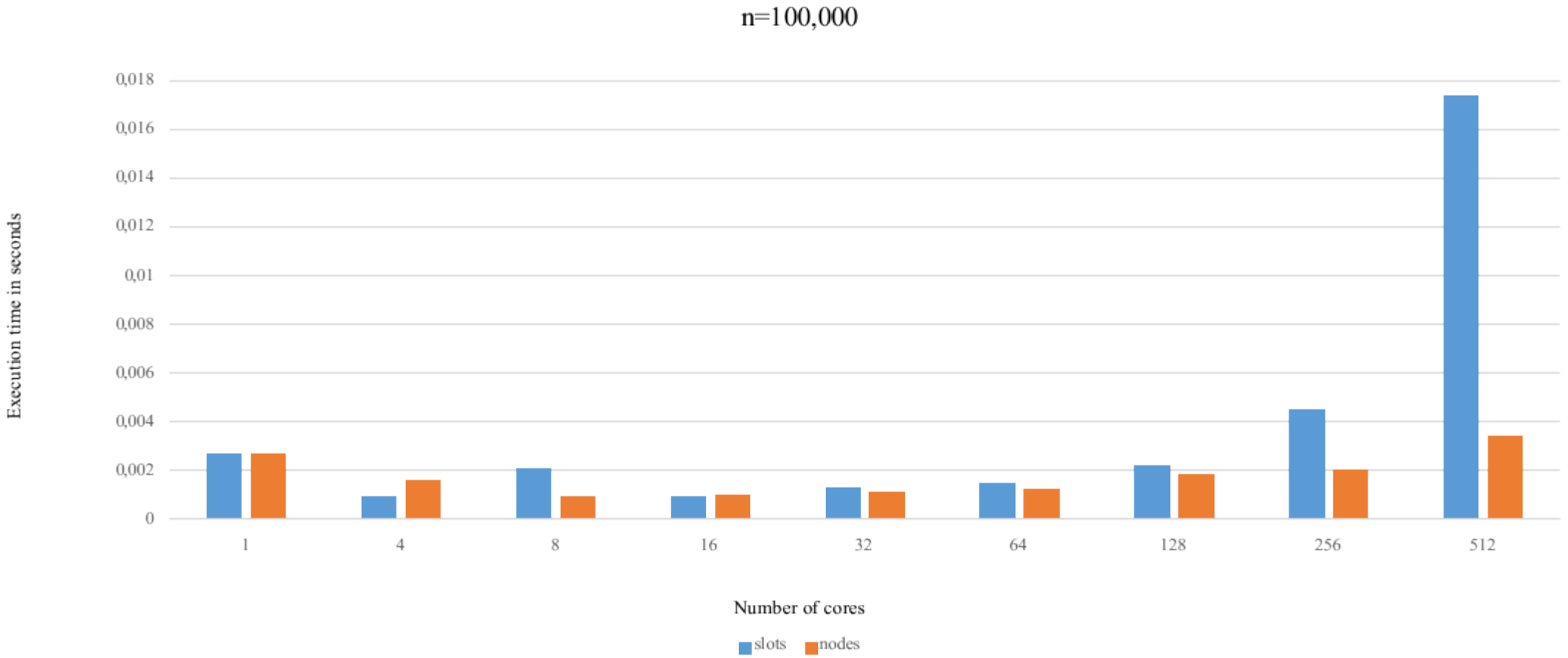
Scaling analysis – Results
n = 1,000



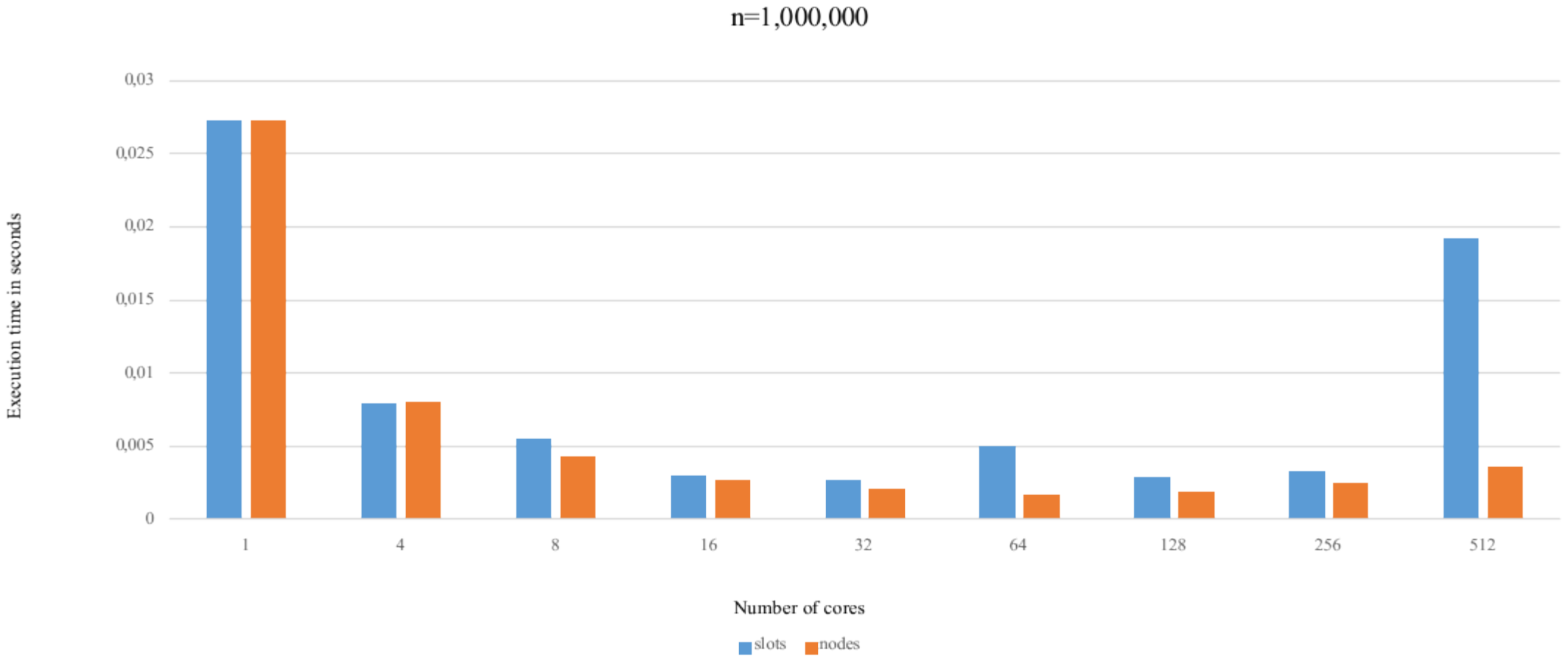
Scaling analysis – Results
n = 10,000



Scaling analysis - Results
n = 100,000

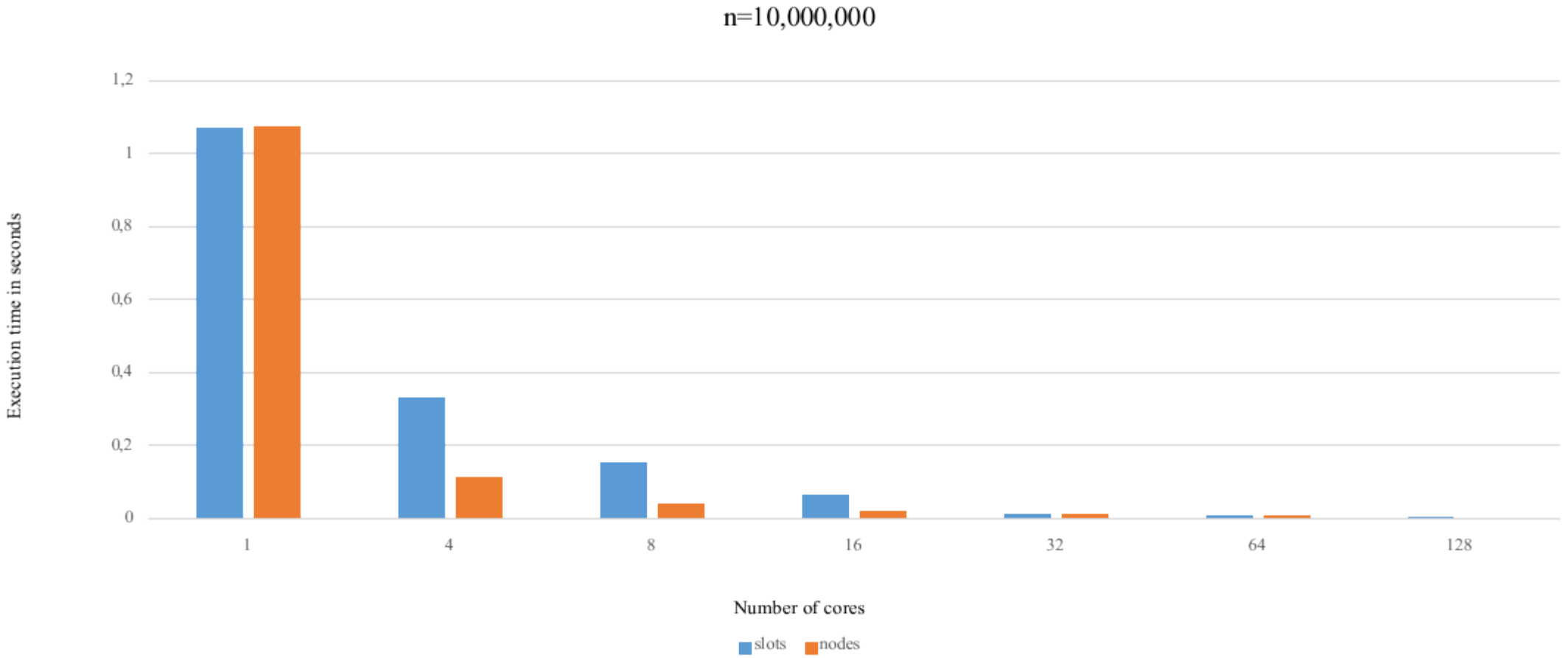


Scaling analysis - Results
n = 1,000,000



Scaling analysis - Results
n = 10,000,000

Memory allocation problems for:
By slot: at 256 cores
By nodes: at 128 cores



Scaling analysis - Results

- Small problems
  - Parallel execution is slower than the execution on a single node
  - Doesn‘t scale well with growing number of cores

- Bigger problems
  - Parallel execution can be faster than the execution on a single node
  - Doesn‘t scale perfectly with growing number of cores
    - After a certain boundary the execution time get worse
- Option by nodes is better than by slots in nearly all cases here

Literature

- http://acc6.its.brooklyn.cuny.edu/~cisc7340/examples/mpisieves16.pdf
- https://upload.wikimedia.org/wikipedia/commons/6/63/Animation_Sieb_des_Eratosthenes.gif