KMP PARALLEL ALGORITHM FOR PATTERN MATCHING

UBIT: rbammidi

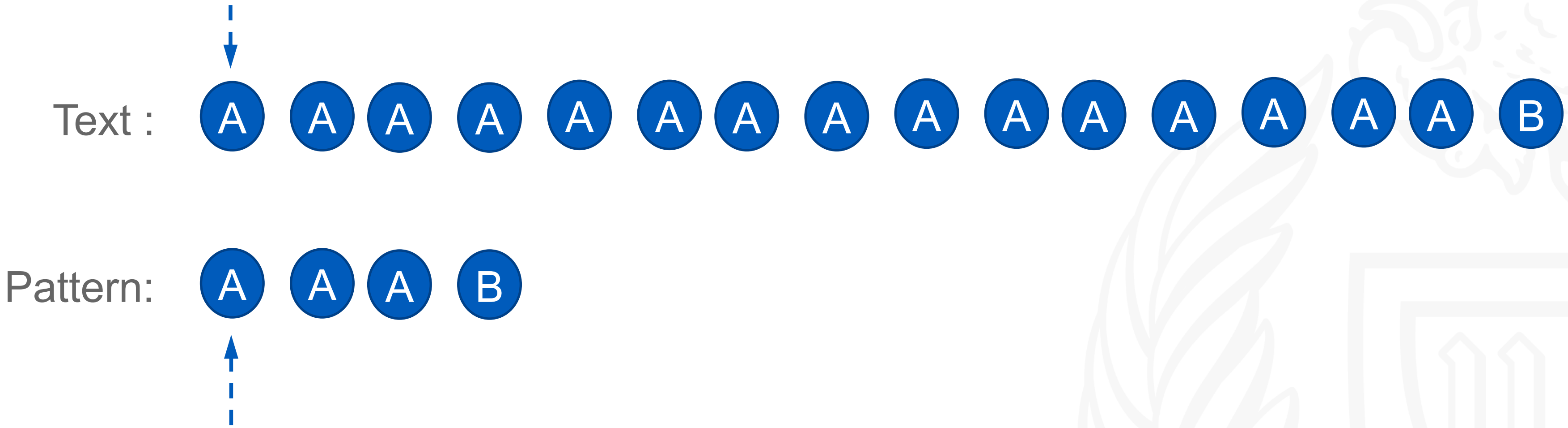
Professor: Dr. Russ Miller

University at Buffalo The State University of New York

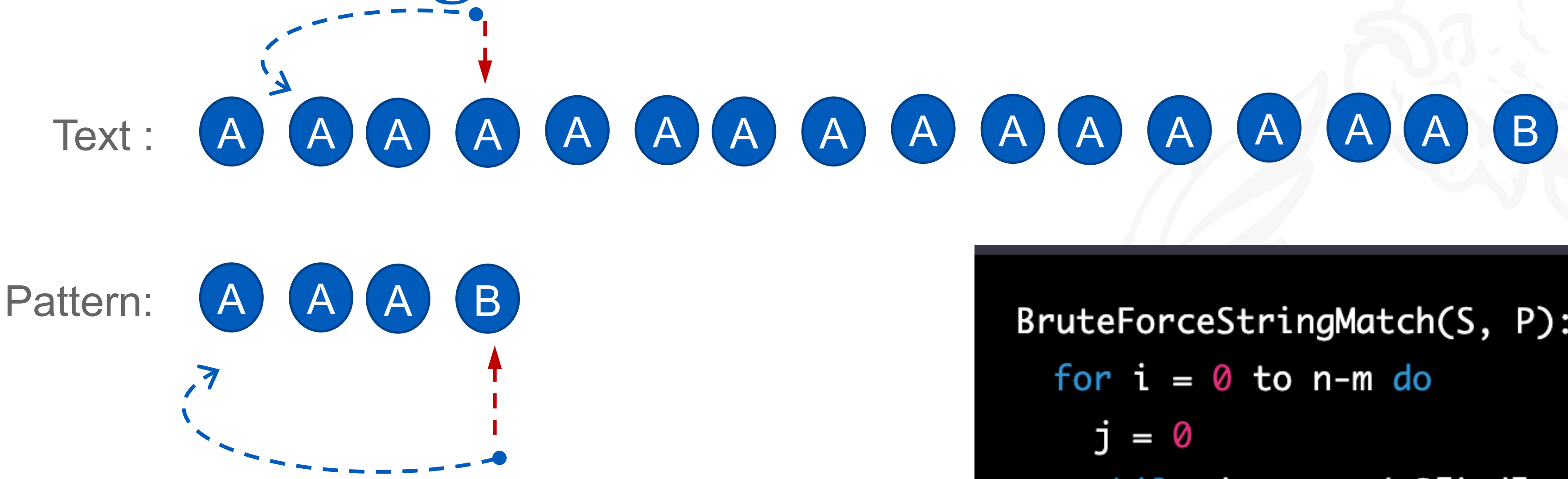
Need of pattern matching

- Pattern matching is used to determine whether source files of high-level languages are syntactically correct.
- Many fingerprint recognition methods are in use to perform fingerprint matching out of which pattern matching approaches is widely used.
- Pattern matching enables users to find the locations of particular DNA subsequences in a database or DNA sequence.
- Searching for word in the large log files dump
- Validating the information received from the client before writing into DB.

Pattern Matching (Naive)



Pattern Matching (Naive)




```
BruteForceStringMatch(S, P):
  for i = 0 to n-m do
    j = 0
    while j < m and S[i+j] == P[j] do
      j = j + 1
    if j == m then
      return i
  return -1
```

n = Length(Text)
m = length(Pattern)

T(n,m)= O(n*m)

Knuth-Morris-Pratt (KMP)

The Knuth-Morris-Pratt (KMP) algorithm is an algorithm that is used to search for a pattern in a given text in O(m + n) time (where m and n are the lengths of pattern and text).

KMP Algorithm ~ 2 Step Process

Step 1: Pre-process the Pattern

step 1.1: Build the LPS table from the pattern

Step 2: Iterate through the Text and Pattern and check for the existence of the pattern in the text

Components and Terminology of KMP Algorithm

In the KMP algorithm, we have two terms, proper prefix and suffix

A proper prefix of the pattern will be a subset of the pattern using only the beginning portion (the first index), or the first few indices of the pattern except the last character

Pattern : a b c d a b c

- a
- a b
- a b c
- a b c d
- a b c d a b c

Components and Terminology of KMP Algorithm

A proper suffix of any pattern would be a subset of the pattern with elements taken only from the right end of the pattern as in, any number of elements, starting from the last character. Taking the first character of the string is not allowed

Pattern : a b c d a b c

- c
- b c
- a b c
- d a b c

Longest prefix that is also a suffix (LPS)

	A	A	B	A	A	A	A	B	A	A	B
LPS[ ]	0	1	0	1	2	2	2	3	4	5	3

LPS[i] represents longest prefix that is also a suffix till i

Takes O(m) time to generate the LPS array

```
int *kmptable(char *pattern, int len)
{
    int k = 0;
    int i = 1;
    int *table = (int *)malloc(len * sizeof(int));
    table[0] = 0;
    while (i < len)
    {
        if (pattern[k] == pattern[i])
        {
            k += 1;
            table[i] = k;
            i++;
        }
        else if (k > 0)
        {
            k = table[k - 1];
        }
        else
        {
            table[i] = 0;
            i++;
        }
    }

    return table;
}
```

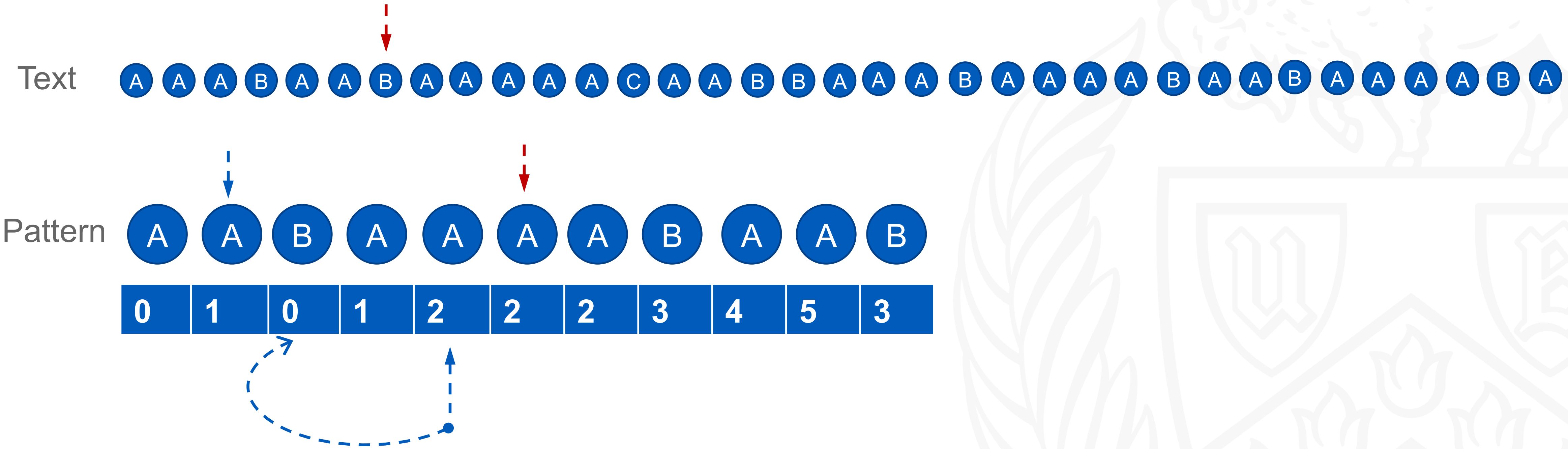
KMP Pattern Matching

Text: A A A B A A B A A A A A C A A B B A A A B A A A A B A A B A A A A B A

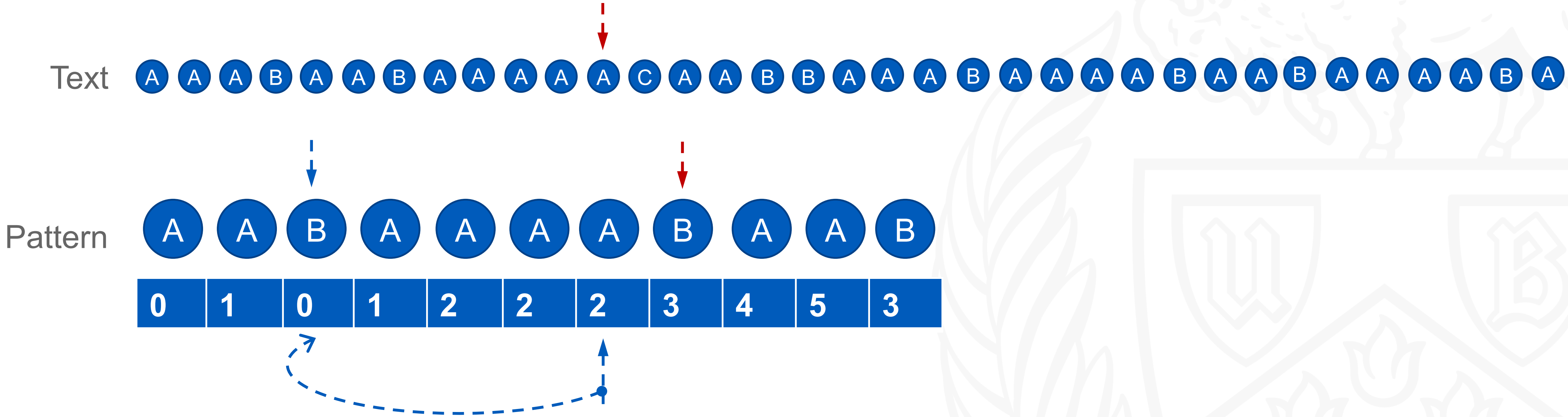
Pattern: A A B A A A A B A A B

0	1	0	1	2	2	2	3	4	5	3

KMP Pattern Matching



KMP Pattern Matching



KMP Pattern Matching

Text: A A A B A A B A A A A A C A A B B A **[A A B A A A A B A A B]** A A A A B A

Pattern: A A B A A A A B A A B

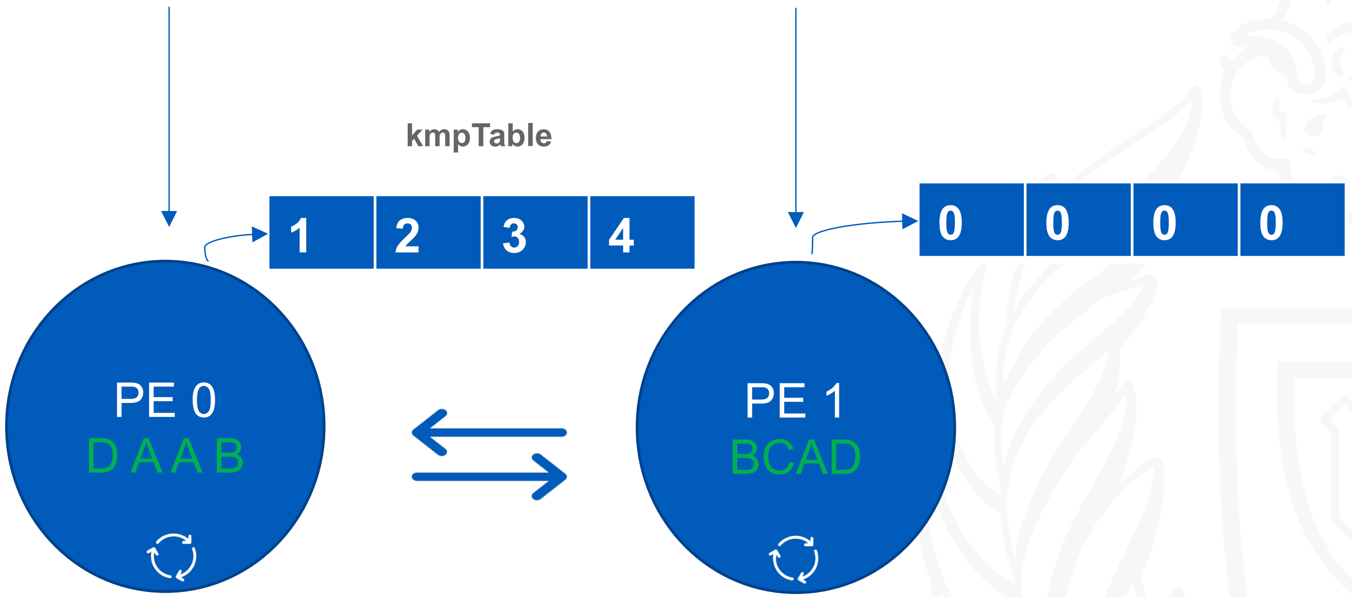
Parallel KMP Algorithm

Components and Terminology of Parallel KMP Algorithm

- KMP Table: kmpTable[i] stores the length of the longest pattern that matches with the text till index i
- Cumulative KMP Table: It holds the cumulative KMP table information from the processor 0 to processor i
- Non-cumulative KMP Table: It holds the non-cumulative kmp table information, which means it doesn't contain the KMP information from processor 0 (partial kmp table)

KMP Table usage

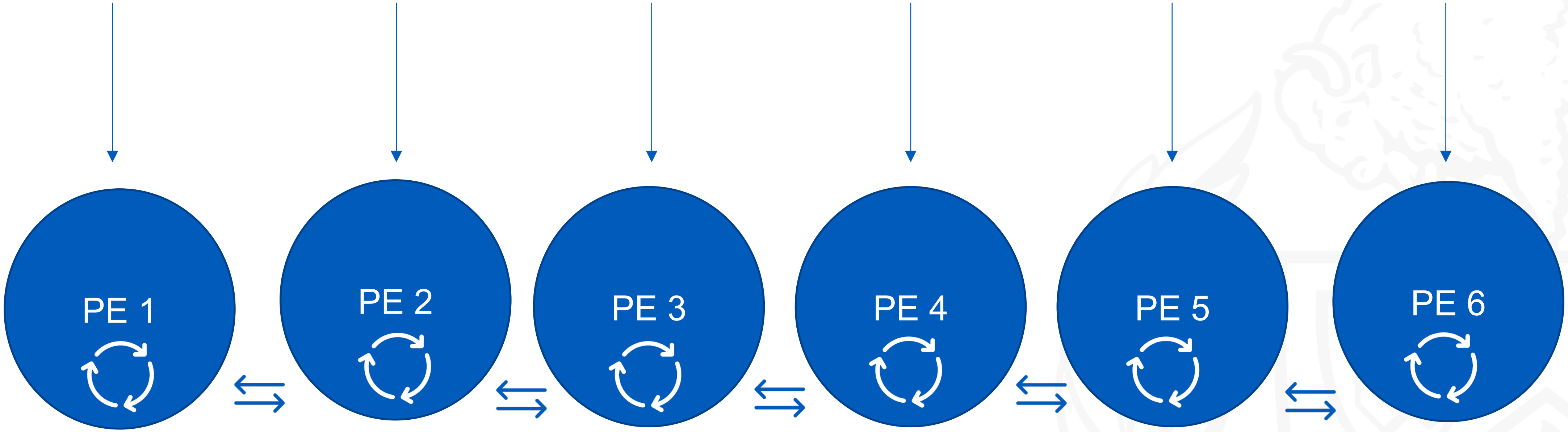
Text: D A A B B C A D



Pattern: **D A A B** B C A D

Pattern: ~~D A A B~~ **B C A D**

Initial Attempt

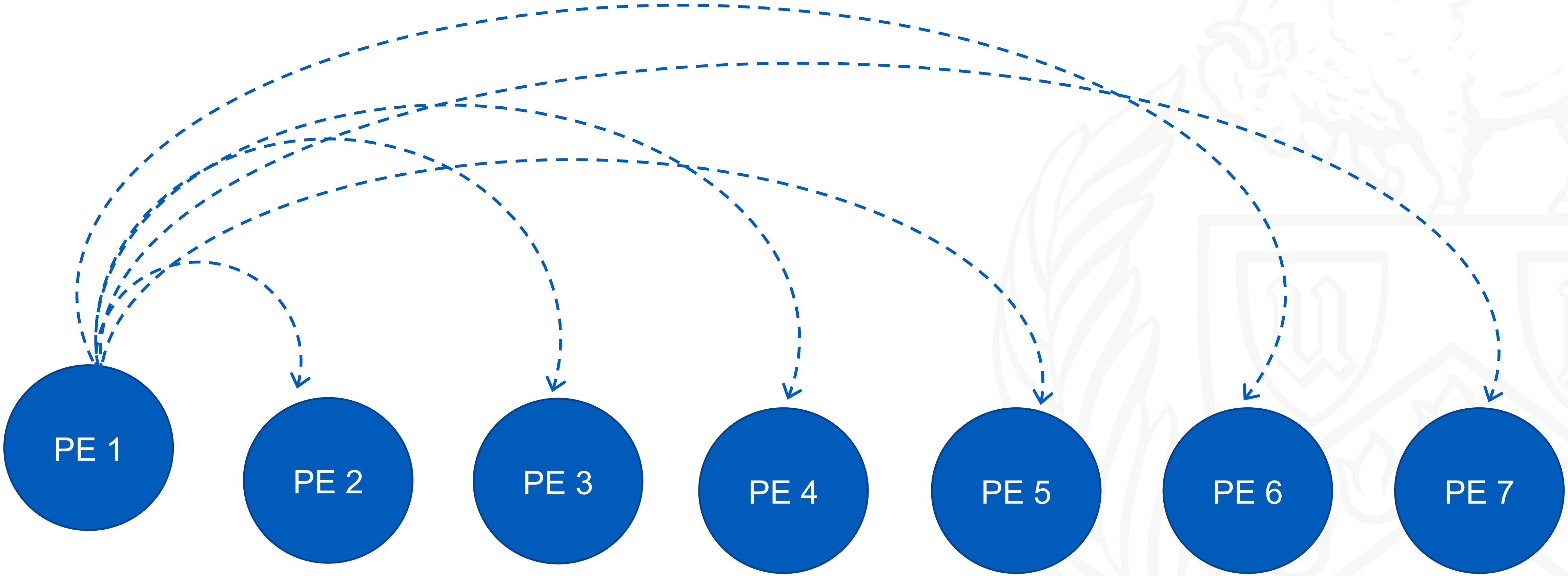


Drawbacks:
Every i^{th} processor has to wait until it receives the parallel KMP table from [0, $i-1$] processors

Parallel KMP Steps

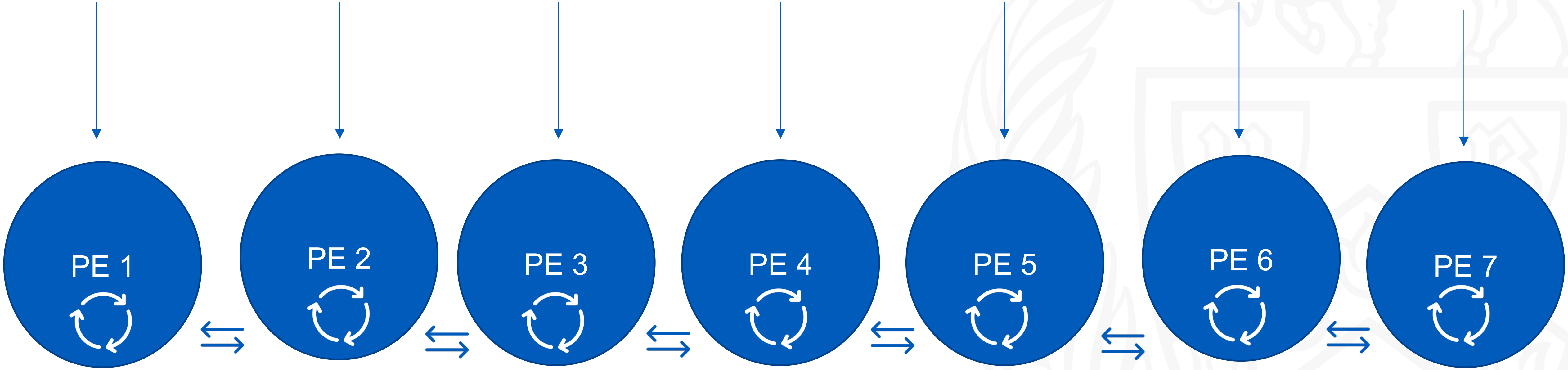
- Split the given text equally of size $(\frac{n}{k})$ $each$ to all the processors ~ Broadcasting
- Each processor executes sequential KMP independently on the given text & pattern
- Every processor checks if the cumulative KMP table is available to receive from its predecessor
- If the cumulative KMP table is not available in the buffer, it receives the non-cumulative KMP table from its preceding processor.
- This process continues till it finds the pattern in the given text.

Broadcasting

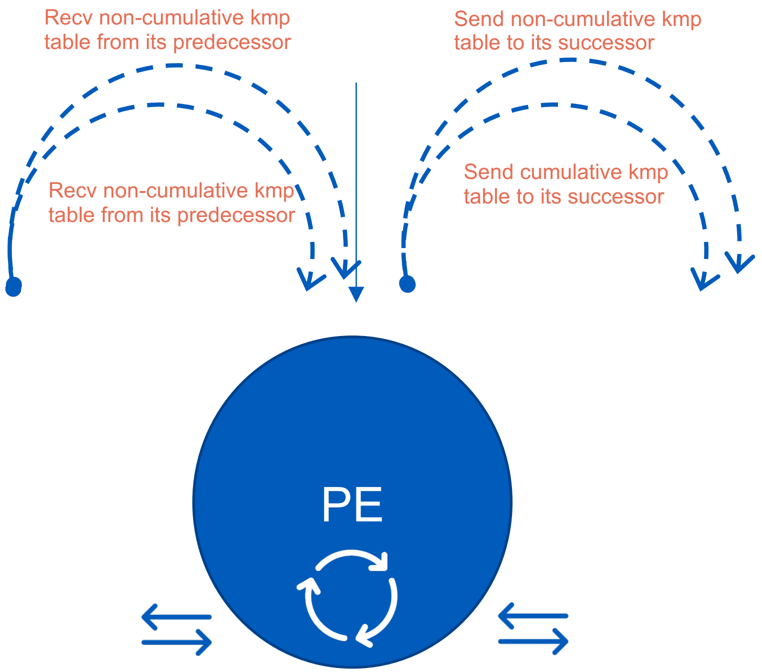


Parallel KMP Visualization

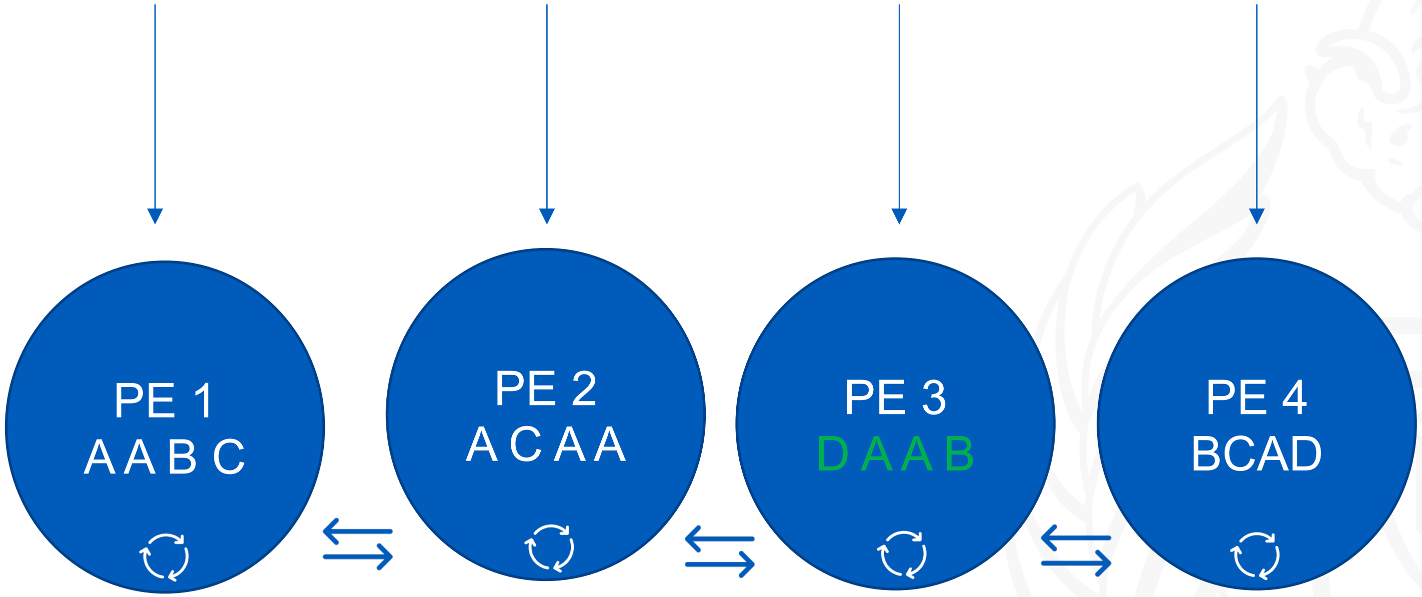
Step1: Process Text & Pattern
Step 2: Async send
Step 3: Async recv



Profiling a typical processor

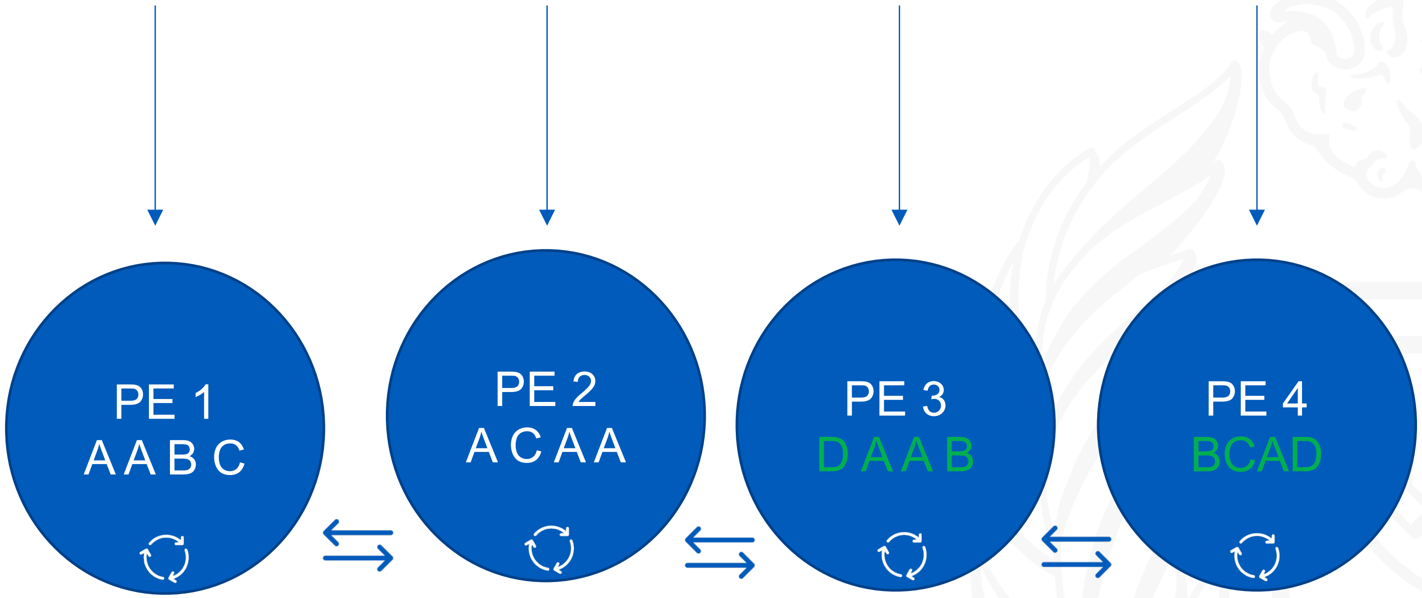


Pattern existence in a single processor



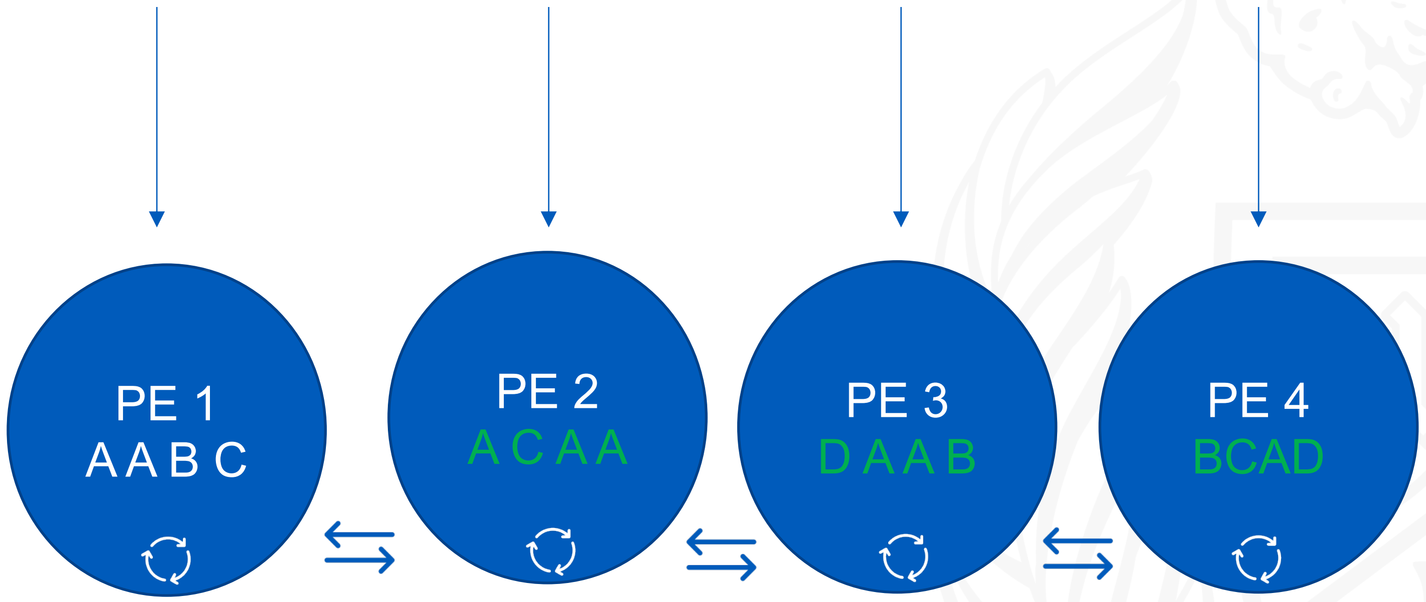
Pattern: D A A B

Pattern existence in two processors



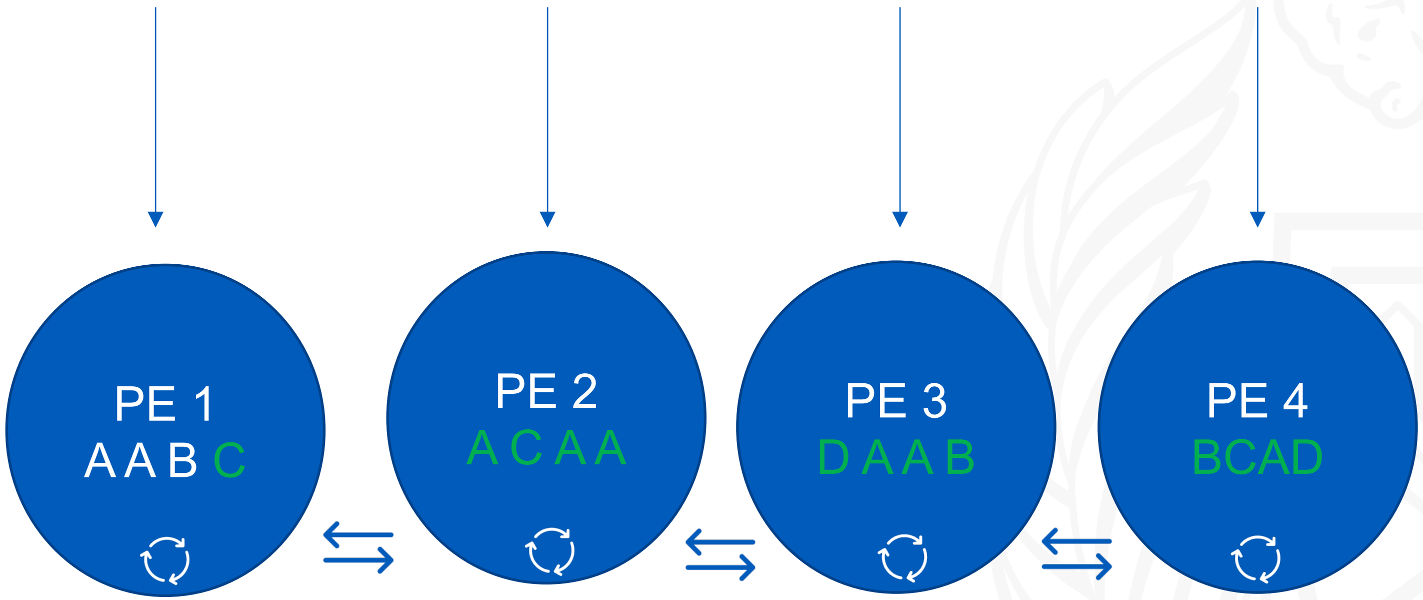
Pattern: D A A B B C A D

Pattern existence in three processors



Pattern: A C A A D A A B B C A D

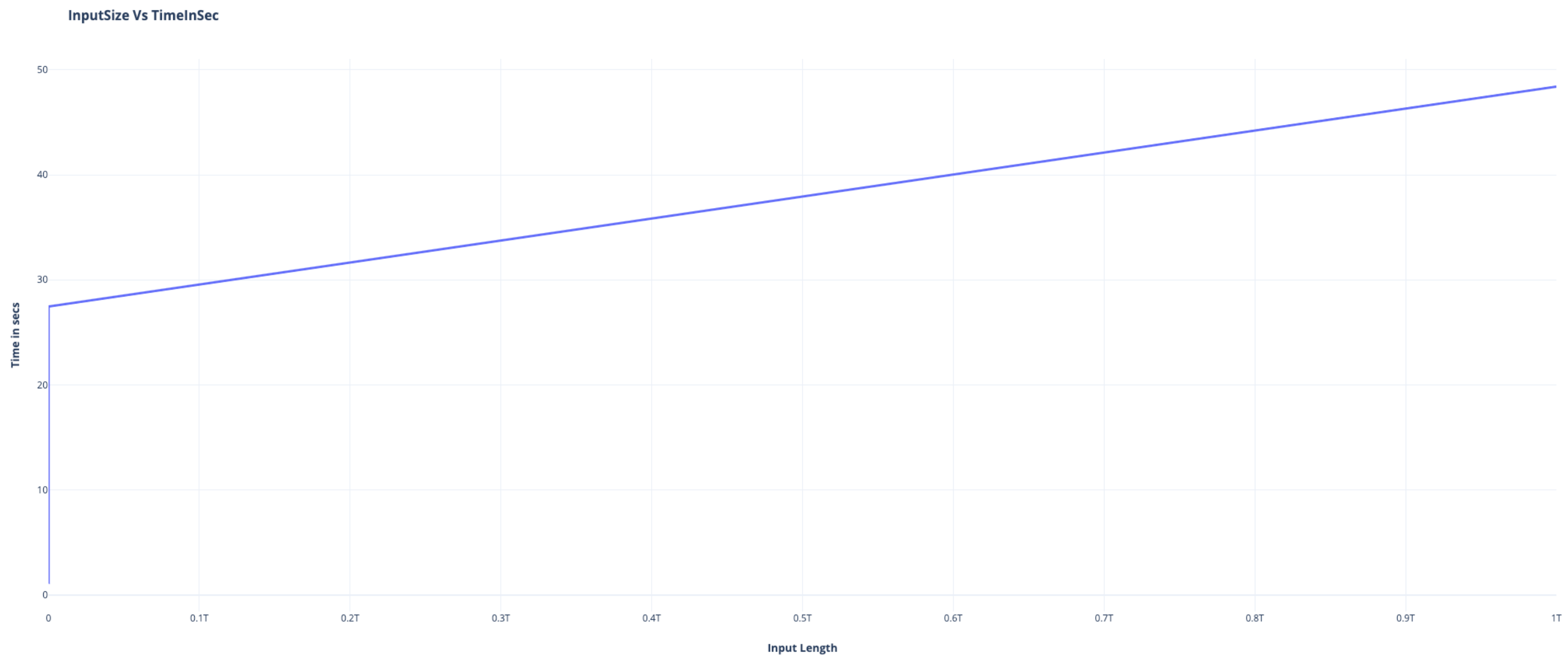
Pattern existence in four processors



Pattern: C A C A A D A A B B C A D

Input Size Vs Time In Secs for Sequential KMP

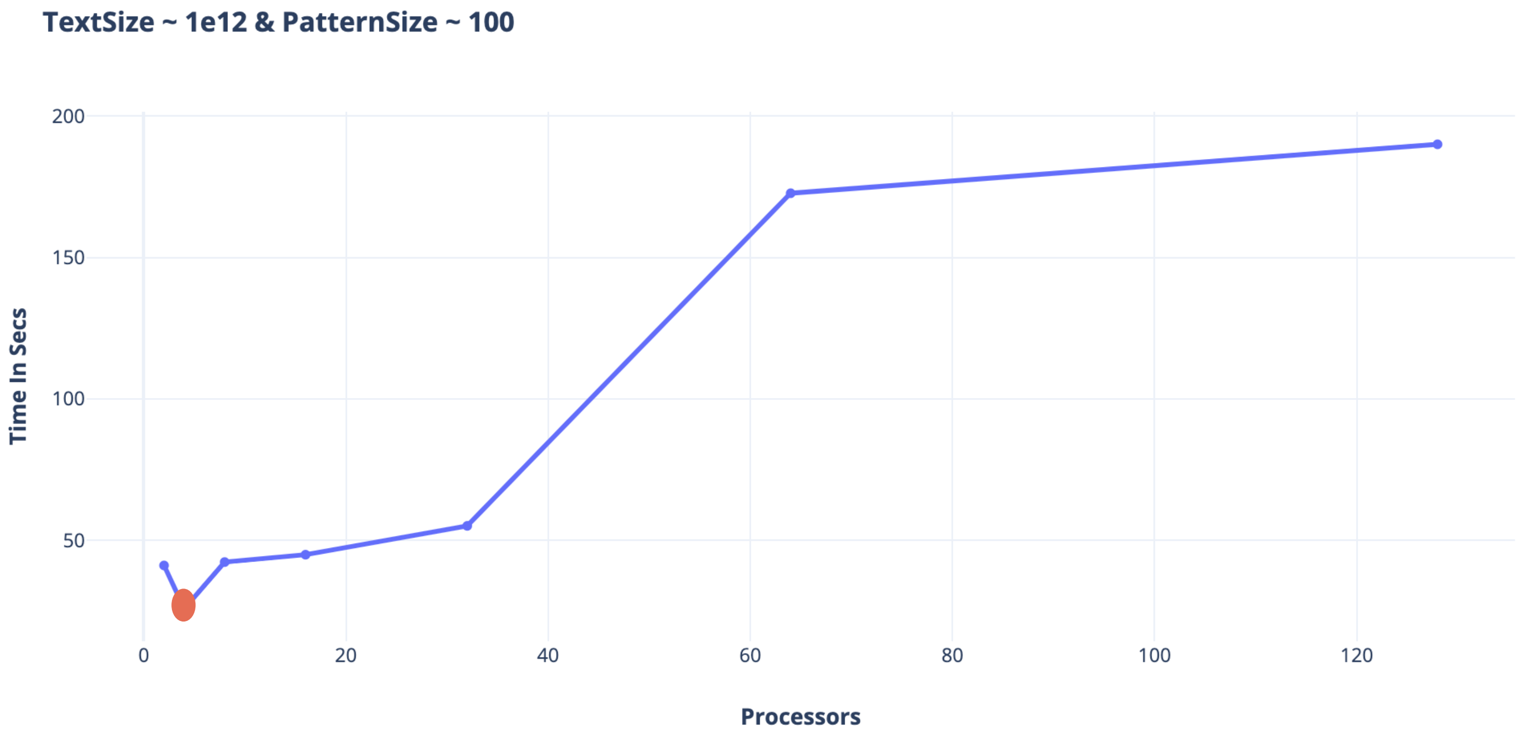
Input Size	Time In Secs
1E3	1.047
1E4	1.259
1E5	3.581
1E6	27.454
1E12	48.4



Processors Vs Run Time Text Size=1e12 & P=33 (slurm)

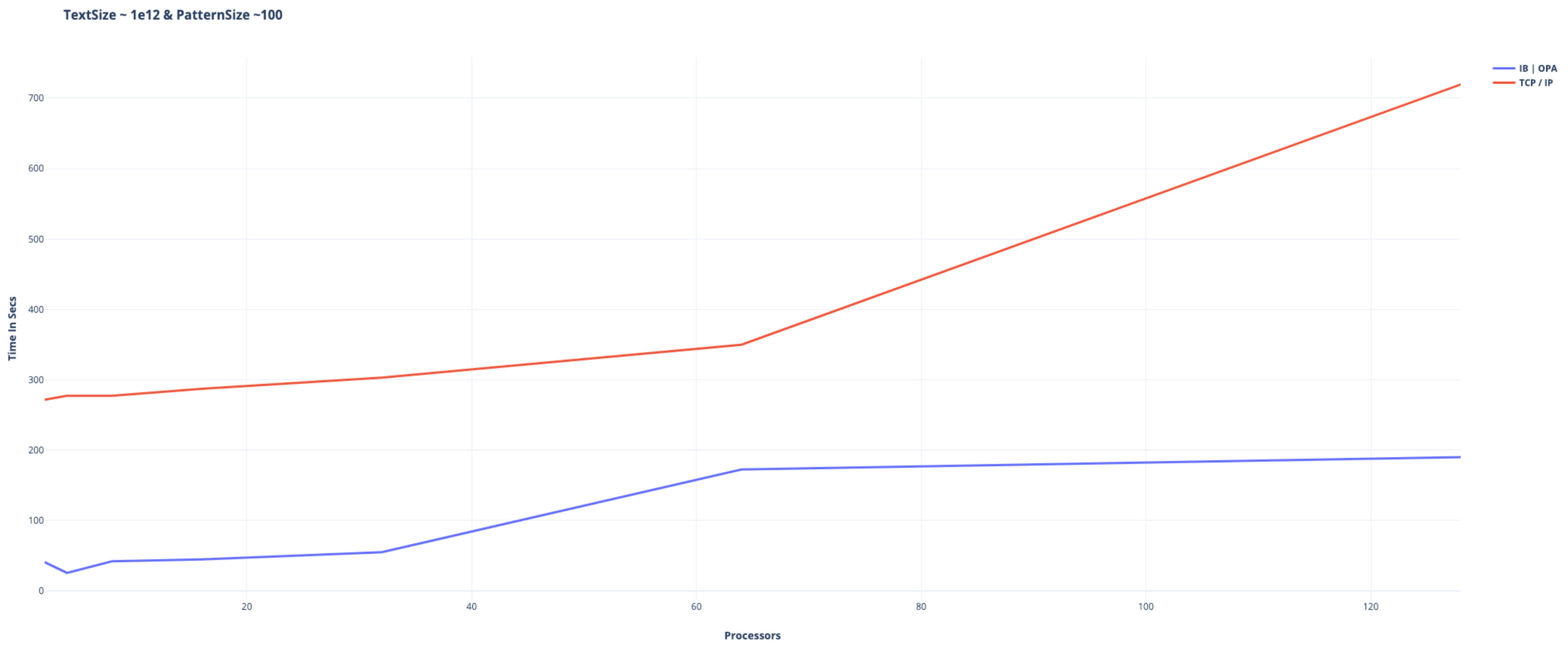
1Processor per node

Processors	Secs
2	41.1
4	25.8
8	42.3
16	44.9
32	55.1
64	172.7
128	190.0



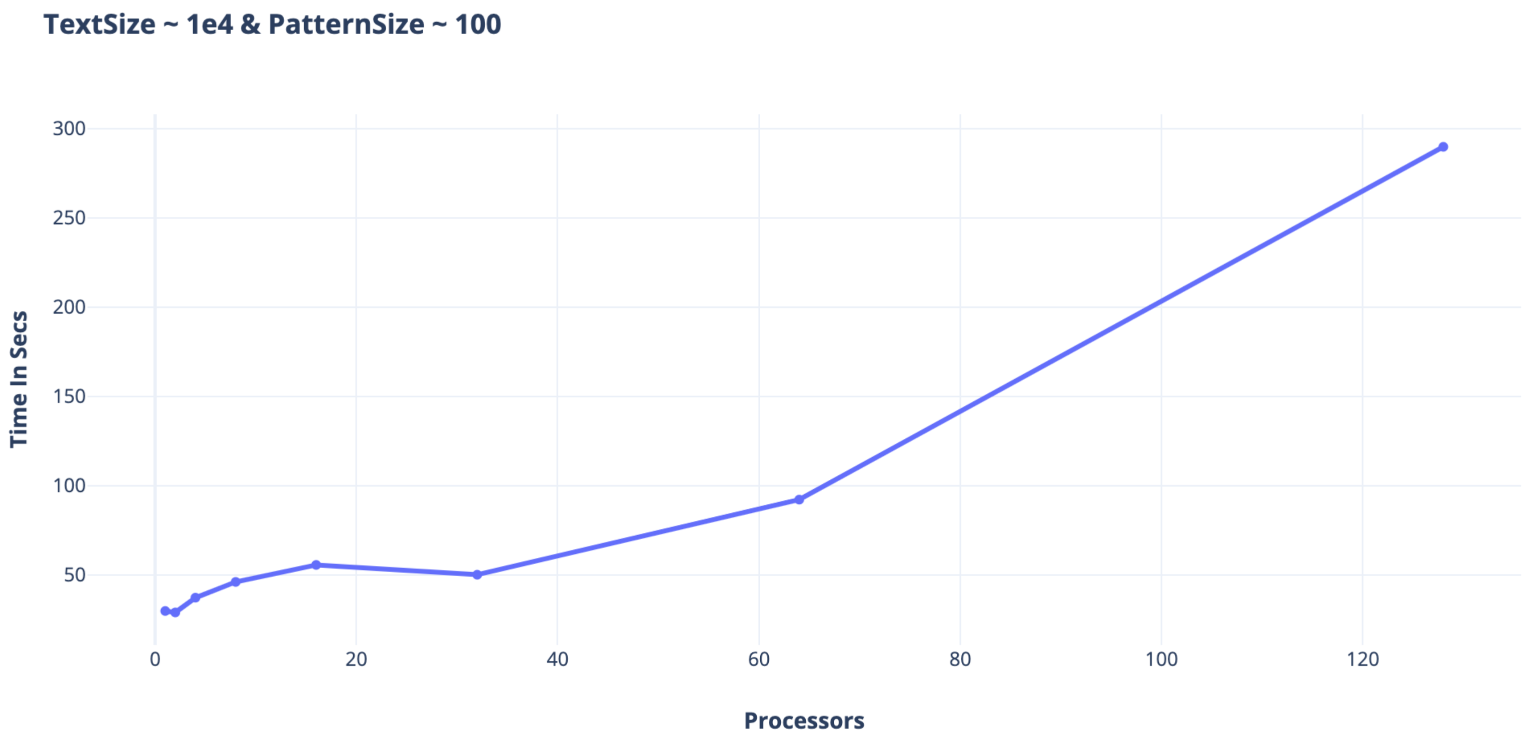
TCP/IP Vs IB|OPA Network Band Performance

Processor	IB	TCP/IP
2	41.1	271.604
4	25.8	277.253
8	42.3	277.377
16	44.9	287.313
32	55.1	302.991
64	172.7	349.8
128	190	719.4



Scale (Constant Data/PE)

Processors	Time in secs	Data Per PE
1	30.821	1e4
2	28.999	1e4
4	37.272	1e4
8	46.078	1e4
16	55.606	1e4
32	50.129	1e4
64	92.168	1e4
128	289.702	1e4



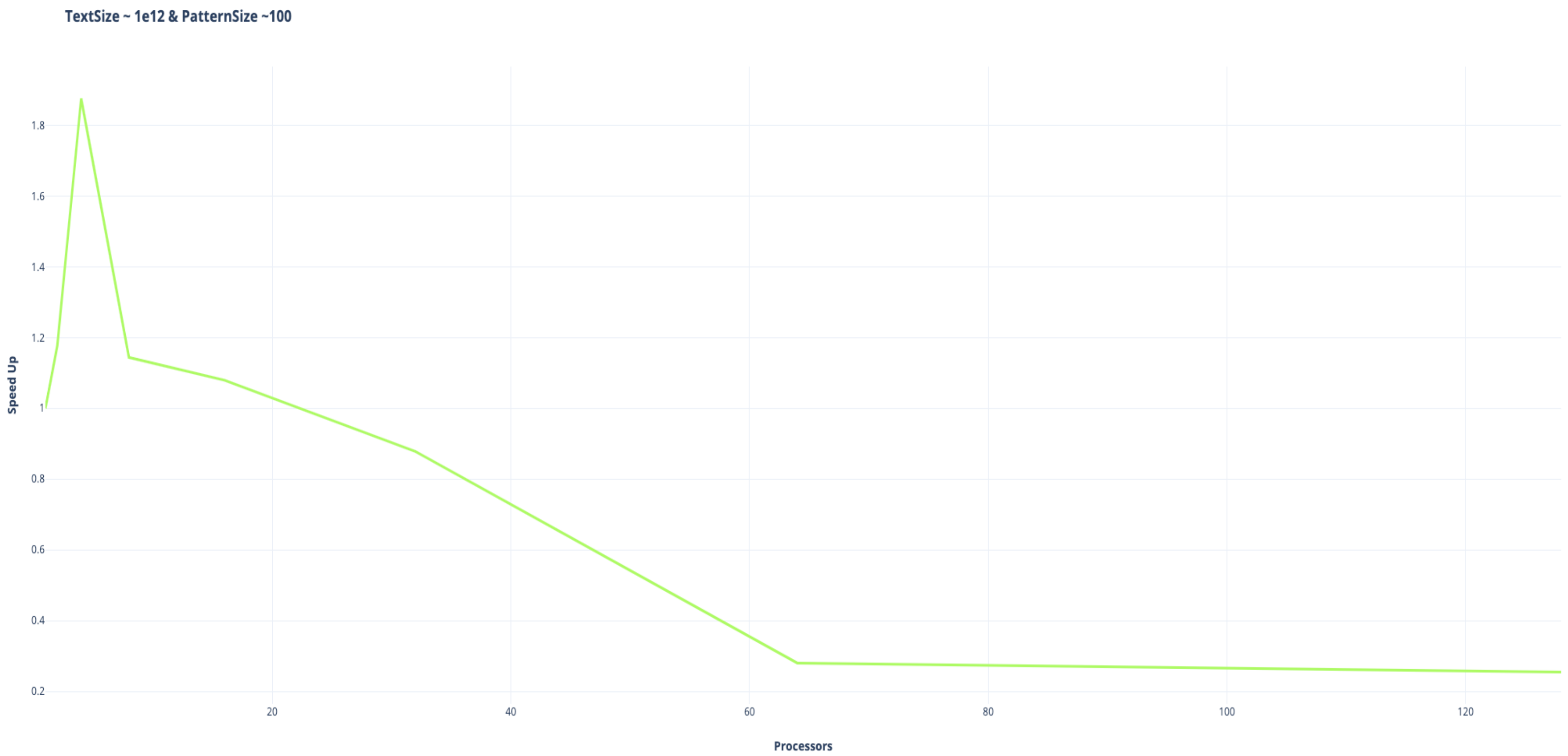
Speed Up Vs Processors

Input text size 1e12, $T_{seq} = 48.478\ secs$

$$SpeedUp = \frac{T_{seq}}{T_{parallel}}$$

1Processor per node

Processors	$T_{parallel}$	Speed Up	Data per PE
1	1	1	1e12
2	41.1	1.17	500000000000
4	25.8	1.87	250000000000
8	42.3	1.14	125000000000
16	44.9	1.07	62500000000
32	55.1	0.87	31250000000
64	172.7	0.28	15625000000
128	190	0.25	7812500000



References

- https://ieeexplore.ieee.org/document/6618720
- https://cmps-people.ok.ubc.ca/ylucet/DS/KnuthMorrisPratt.html
- https://buffalo.app.box.com/s/nqj3neyt2w1dtb3gix6zxqx5gcc9x30n
- http://koreascience.or.kr/article/JAKO201814955686557.page
- https://ieeexplore.ieee.org/document/8599534

Thank You

Questions?