Optimal Parallel Algorithm for Periods

Alberto Apostolico, Dany Breslauer, Zvi Galil, 1991

POLITECNICO
MILANO 1863

Carminati Andrea
ID: 877886
andrea1.carminati@mail.polimi.it

Main Goal

Finding all the periods of a string. The period of a string can be computed by previous efficient parallel algorithms only if it is shorter than half of the length of the string.

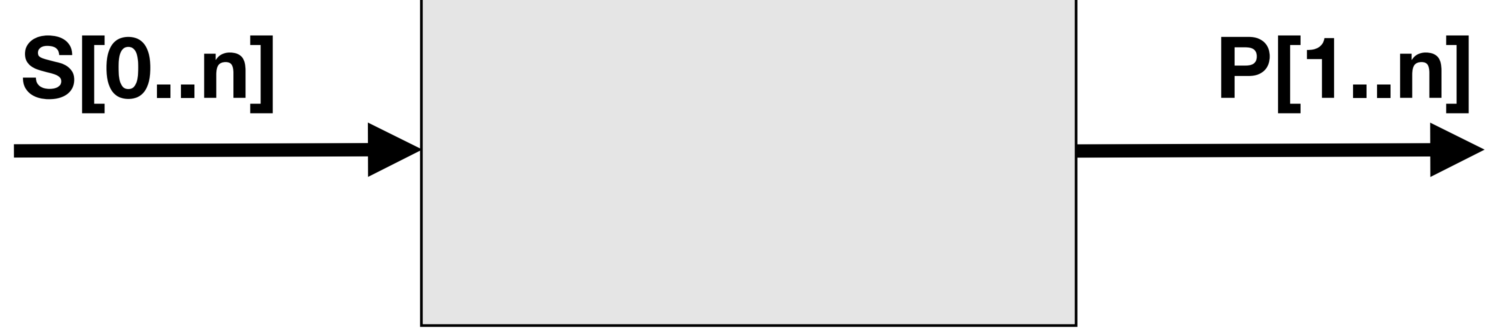
This new algorithm computes all the periods, even if they are longer, in optimal **O(log log n) time**.

The scope of this project is to develop in **OpenMP** the optimal **CRCW-PRAM algorithms** presented in the paper that solve the problem of finding all periods of a string.

The solution is **the fastest possible optimal parallel algorithms** for these problems over a general alphabet.

Finding all periods

We describe an algorithm that **given a string S[0..n]** will **compute all the periods of S**. The output of the algorithm will be a Boolean array **P[1..n] such that P[i] = true if and only if i is a period of S**.

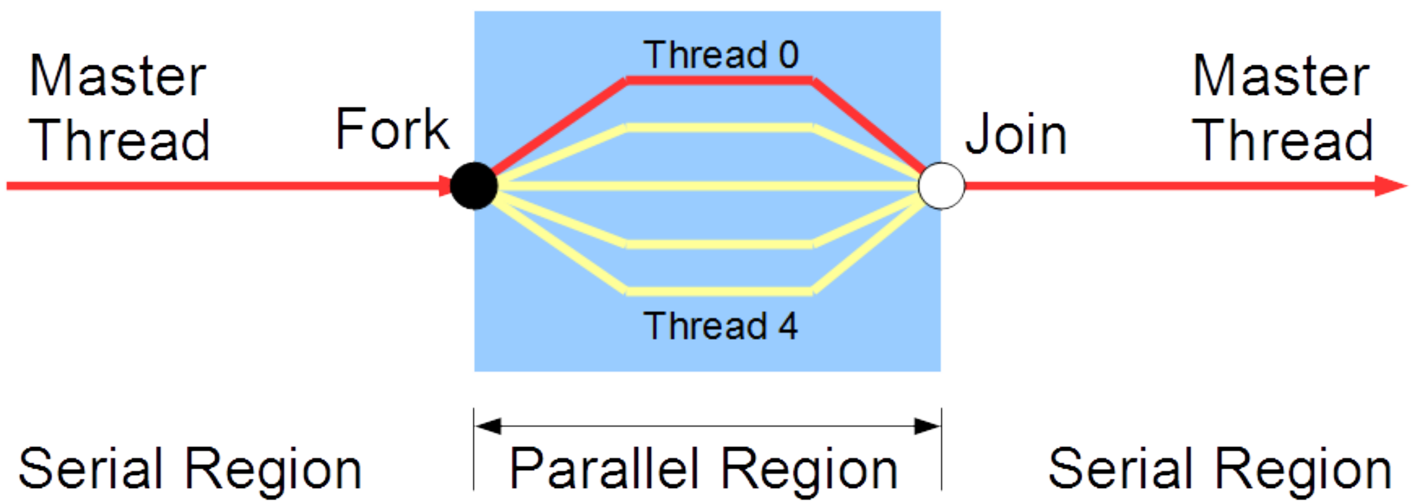


Thesis

" There exists an algorithm to compute P[1..n] that takes O(log log n) time using n/(log log n) processors. "

The parallelization

The algorithm will proceed in **independent stages which are theoretically computed all simultaneously.**



The number of stages to parallelize depends on the length of the input string.

We will test the program with different number of core.

A Single Stage

In **stage number η** we will compute only $P[n-l(\eta)+1 .. n-l(\eta+1)]$.

The sequence $\{l(\eta)\}$ is a decreasing sequence defined as:

- **$l(0)=n$**
- **$l(\eta+1)=$ floor$(2/3*l(\eta))$**

Where $0<=\eta<m$ and m is the smallest integer for which $l(m)=0$

NB: stage is assigned to compute a disjoint part of the output array P and the entire array is covered.

1° String Matching

We start to call a string matching algorithm to **find all** occurrences of **S[0 .. l(η+1)]** in **S[n-l(η)+1 .. n]**.

Let qi, where i = 1 .. **r**, denote the indices of all these occurrences.
(all indices are in the string S[0 .. n], thus n-l(η)<qi<n-l(η+1)).

If there where no occurrences String S has no period in the range computed and **all entries of P[n-l(η)+1..n-l(η+1)] are set to false**.

If there was only one occurrences of S[0 .. l(η+1)] in S[n-l(η)+1..n], **it can be verified to be a period** in O(lη) operation.

Indices of string

Stage η	ln	S[0 … l(η+1)]	Length	S[n-l(η)+1 … n]	Length
0	44	S[0 … 29]	30	S[1 … 44]	44
1	29	S[0 … 19]	20	S[16 … 44]	29
2	19	S[0 … 12]	13	S[26 … 44]	19
3	12	S[0 … 8]	9	S[33 … 44]	12
4	8	S[0 … 5]	6	S[37 … 44]	8
5	5	S[0 … 3]	4	S[40 … 44]	5
6	3	S[0, 1, 2]	3	S[42, 43, 44]	3
7	2	S[0, 1]	2	S[43, 44]	2
8	1	S[0]	1	S[44]	1

Search S[0 … l(η+1)] in this S[n-l(η)+1 … n]

2° String Matching

Otherwise, if there are r>1 recurrences we continue with **another call to a string matching algorithm to find all occurrences of S[0 .. l(n+1)] in S[0 .. l(n)-1]**.

Let pi, i=1..k, denote the indices of all there occurrences.

NB: the string to find is the same as before but look for another part of the text.

After this step we have to check that the following lemmas hold…

Indices of string

Stage η	In	S[0 … l(η+1)]	Length	S[0 .. l(n)-1]	Length
0	44	S[0 … 29]	30	S[0 … 43]	44
1	29	S[0 … 19]	20	S[0 … 28]	29
2	19	S[0 … 12]	13	S[0 … 18]	19
3	12	S[0 … 8]	9	S[0 … 11]	12
4	8	S[0 … 5]	6	S[0 … 7]	8
5	5	S[0 … 3]	4	S[0 … 4]	5
6	3	S[0, 1, 2]	3	S[0,1, 2]	3
7	2	S[0, 1]	2	S[0, 1]	2
8	1	S[0]	1	S[0]	1

Search S[0 … l(η+1)] in this S[0 .. l(n)-1]

Lemmas for Repeating Periods

- **Lemma 1**: If a string of length m has two periods of length p and q and p+q<=m, then it has also a period of length gcd(p,q). This structure enable us to proceed efficiently to test which of the qi's is actually a period of s.

- **Lemma 2**: If a string A[1 .. l] has period p and occurs only at positions p1<p2<..<pk of a string B[1 .. ceil(3/2*l)], then the pi's form an arithmetic progression with difference p.

- **Lemma 3**: The sequence {pi} and {qi} form an arithmetic progression with difference P, **where P is the period S[0 .. l(n)+1]**

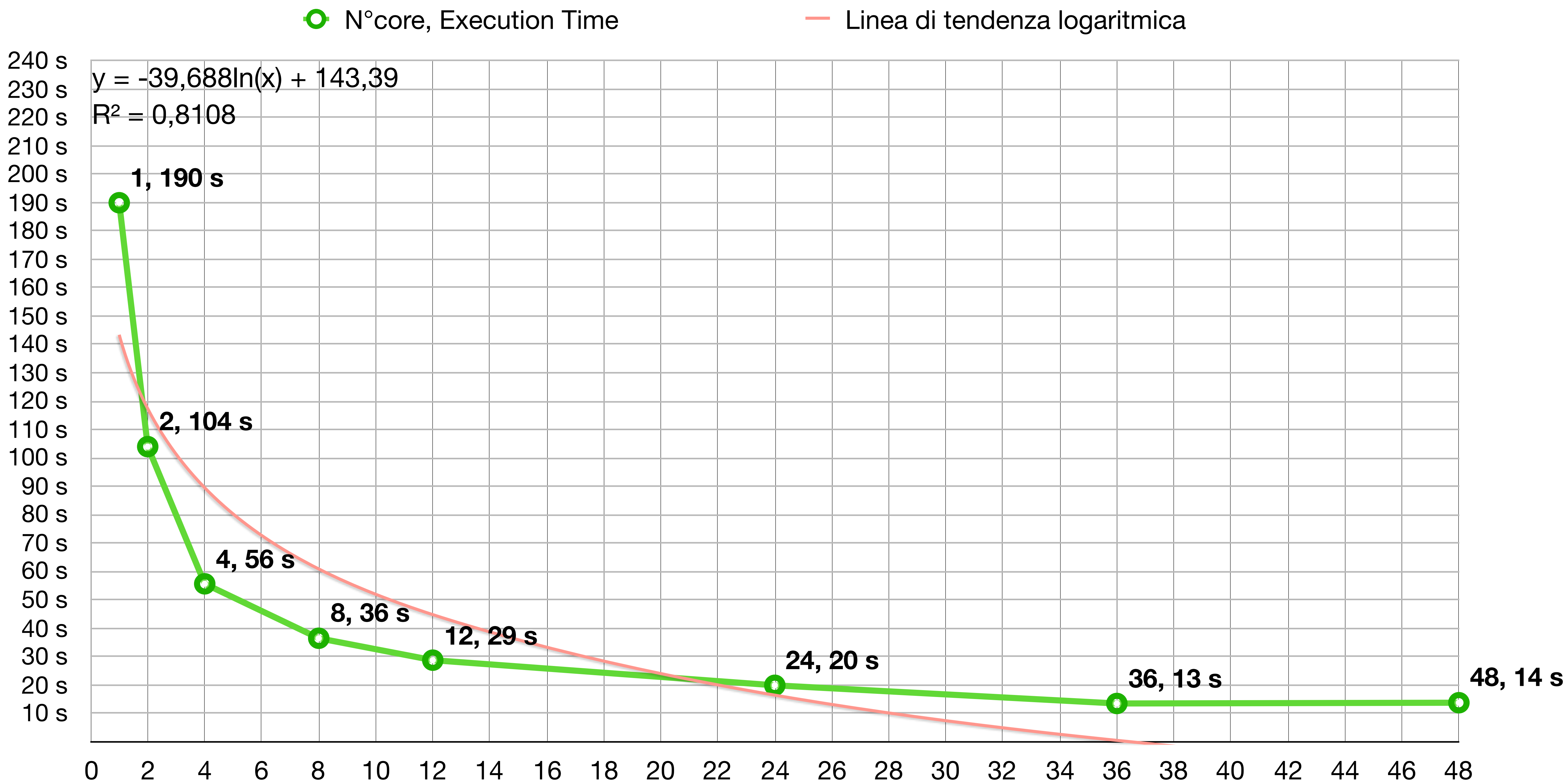
Lemmas for Mismatching and Overflow Periods

- **Lemma 4**: If k (n° of pi recurrences) is less than r than qi is not a period of S[0 .. n] for 1<=i<=r-k

- **Lemma 5**: If S[q(r)+P .. N]!=*S[0 .. N-qr-P] t*hen, S has at most one period in the range computed by this stage. This only possible period may exist if *k<r a*nd it is q(r-k+1).

- **Lemma 6** (an overflow): If S[qr+P ..n) = S[0..n-qr-P] then:
    A. If r<k then q1,..,qn are periods of S.
    B. If r>=k then q(r-k+2), … ,q(r) are periods of S. In this case q(r-k+1) can also be a period of S.

Performances



Dataset: value generated from cosine function repeated for 20 periods

Total number of char: 105300 => 26 stages