PPAM 2019

Bialystok, Poland, September 8-11, 2019

13th INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING AND APPLIED MATHEMATICS

Fifth workshop on Models, Algorithms and Methodologies for Hybrid Parallelism in new HPC Systems

An adaptive strategy for dynamic data clustering with the K-means algorithm

Giuliano Laccetti[1], Marco Lapegna[1], Valeria Mele[1], Diego Romano[2]

1) depart. Mathematics and Applications – Univ. of Naples Federico II
2) Institute for high performance computing and networking of the CNR.

Given

- an integer K
- a set $S = \{ s_n \in R^d , n = 1,..,N\}$ of N vectors in the d-dimensional real space

the **K-means algorithms** is aimed to **collect the items of S in K subset** (called clusters) of a partition $P_K = \{C_k \subset S, k = 1,..,K \}$ **on the basis of their similarity**

the traditional description of the K-mean algorithm is

1. subdivide the N items in K arbitrary clusters, each of them with N_k items
2. compute the center c_k of the clusters with the vector operation
$$c_k = \frac{1}{N_k} \sum_{n=1}^{N_k} s_n$$
3. for each s_n find the cluster $C_{\bar{k}}$ that minimize the euclidian distance from its center
$$min||s_n - c_k||_2 , \qquad k = 1,..,K$$
4. reassign each s_n to the new cluster $C_{\bar{k}}$
5. repeat steps 2 - 4 until there is no change

main problems of the K-means algorithm

- The value of K is an input data and it must be fixed before the execution (not true for several applications)
  - K too small : dissimilar items can be grouped in the same cluster
  - K too large : similar items can be assigned to different clusters

- the result strongly depends on the initial assignment of the elements to the clusters (convergence to a local optimum)

execute the algorithm several times with increasing values of K,
and some quality index is used to choose a "good solution".

$$\text{RMSSTD} = \sqrt{\frac{\sum_k \sum_{s_n} \|s_n - c_k\|^2}{d(N-K)}}$$

root-mean-square standard deviation

is a measure of the homogeneity of the clusters
of the resulting partition.

- Large values of RMSSTD indicates that the clusters are not homogeneous.
- Usually RMSSTD decreases when K increases
- A growth of RMSSTD indicates that a homogeneous cluster has been splitted

Algorithm 1: dynamic K-means algorithm

1) Set the number of clusters $K = 0$

2) <u>repeat</u>

2.1) Increase the number of clusters $K = K+1$

2.2) Assign randomly the N elements $s_n \in S$ to arbitrary K clusters C_K each of them with N_k items

2.3) <u>repeat</u>

2.3.1) Compute the center c_k of each clusters C_k

2.3.2) For each $s_n \in S$ find the cluster $C_{\bar{k}}$ minimizing the Euclidean distance $\|s_n - c_k\| \quad k = 1, .., K$

2.3.3) Reassign the elements s_n to the new clusters

<u>until</u> (no change in the reassignment)

2.4) update RMSSTD

<u>until</u> (RMSSTD starts to grow or it is smaller than a given threshold)

remark

the computational cost of the step

2.3.3) Reassign the elements s_n to the new clusters

strictly depends on the initial distribution of the elements s_n in the K clusters C_K

An unsuitable initial assignment can result in a huge number of movement of the elements s_n among the clusters C_K

Our method is designed to reduce the movements of the elements among the clusters, with the aim of achieving a trade-off between a good initial distribution with a reasonable computational cost.

main idea of our approach

to use, at each iteration of the outer iterative structure of the Algorithm 1, the **partition of the elements already defined in the previous iteration,** <u>**working only on the clusters with the more dissimilar elements**</u>

To this aim, let consider the standard deviation of the elements $s_k \in C_k$

$$\sigma_k = \sqrt{\frac{1}{N_k - 1}\sum_{n=1}^{N_k}(s_n - c_k)^2}$$

The value of $\boldsymbol{\sigma_k}$ **can be used to measure the similarity** of the elements in C_k

Greater the value σ_k, farther to the center c_k are the elements of C_k ,
so that it is composed by dissimilar elements.

main idea of our approach

at each iteration, the initial distribution of the elements in the clusters is defined by **splitting in two subset C_α and C_β only the cluster $\widehat{C_{K-1}}$ with the largest standard deviation in the previous iteration**

More precisely:

- $K = 1$ $\quad P_1 = \{C_1\}$ where $C_1 = S$
- $K > 1$ $\quad P_K = P_{K-1} - \{\widehat{C_{K-1}}\} \cup \{C_\alpha, C_\beta\}$

This strategy is based on the **assumption** that, at a given iteration K, very similar items have been already grouped in compact clusters with small values for the standard deviation σ_k , which therefore does not require an assignment to a new cluster.

Algorithm 2: adaptive K-means algorithm

implementation of the adaptive strategy

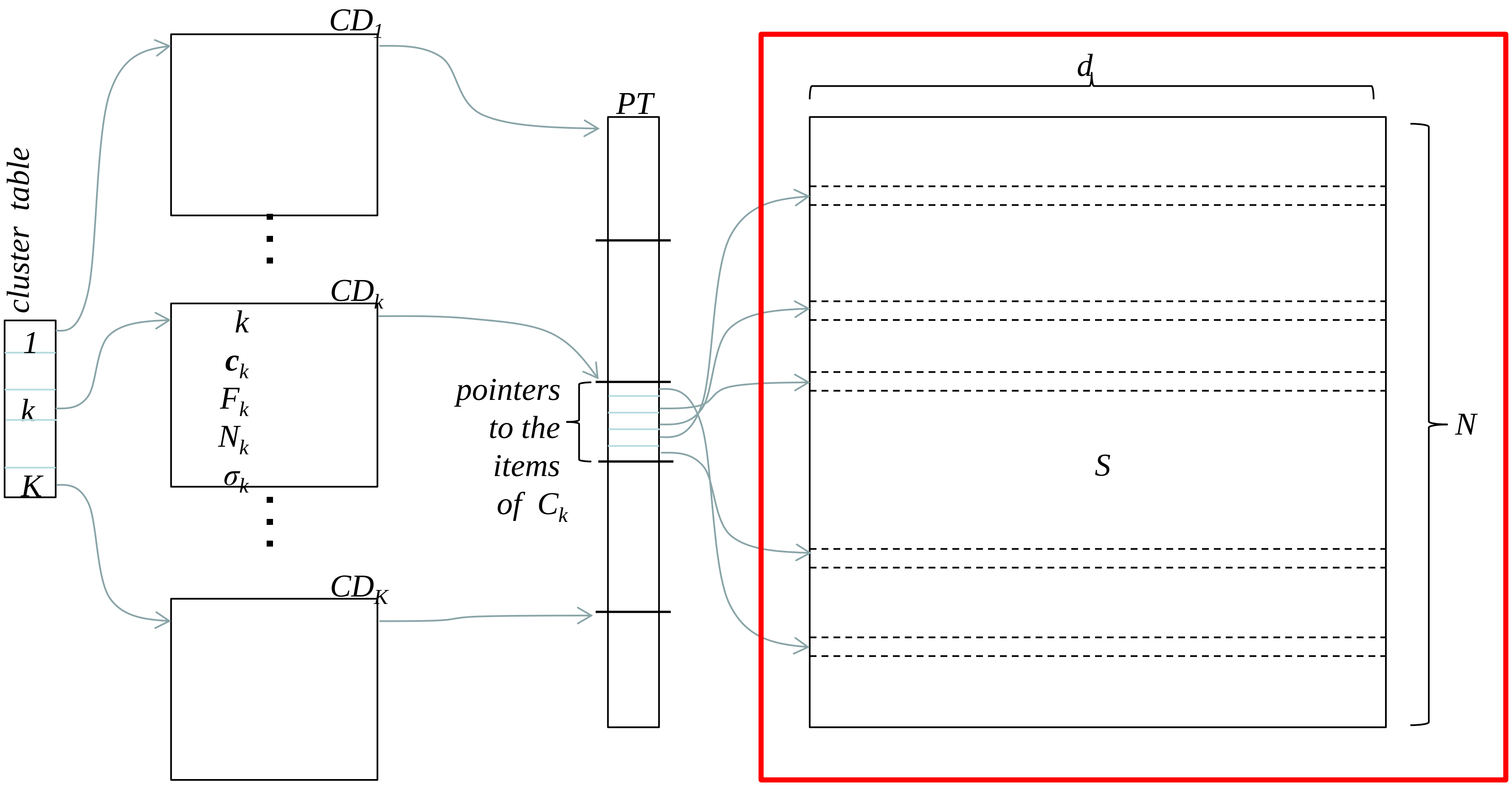
1) Set the number of clusters $K = 0$
2) <u>repeat</u>
   2.1) Increase the number of clusters $K = K+1$
   2.2) find the cluster $\widehat{C_{K-1}} \in P_{K-1}$ with the largest standard deviation
   2.2) Define the new partition $P_K = P_{K-1} - \{\widehat{C_{K-1}}\} \cup \{C_\alpha, C_\beta\}$

   kernel of the algorithm

   2.4) <u>repeat</u>
      2.4.1) Compute the center c_k of each clusters C_k
      2.4.2) For each $s_n \in S$ find the cluster $C_{\bar{k}}$ minimizing the Euclidean distance $||s_n - c_k||$ $k = 1,..,K$
      2.4.3) Reassign the elements s_n to the new clusters
   <u>until</u> (no change in the reassignment)
   2.5) update RMSSTD

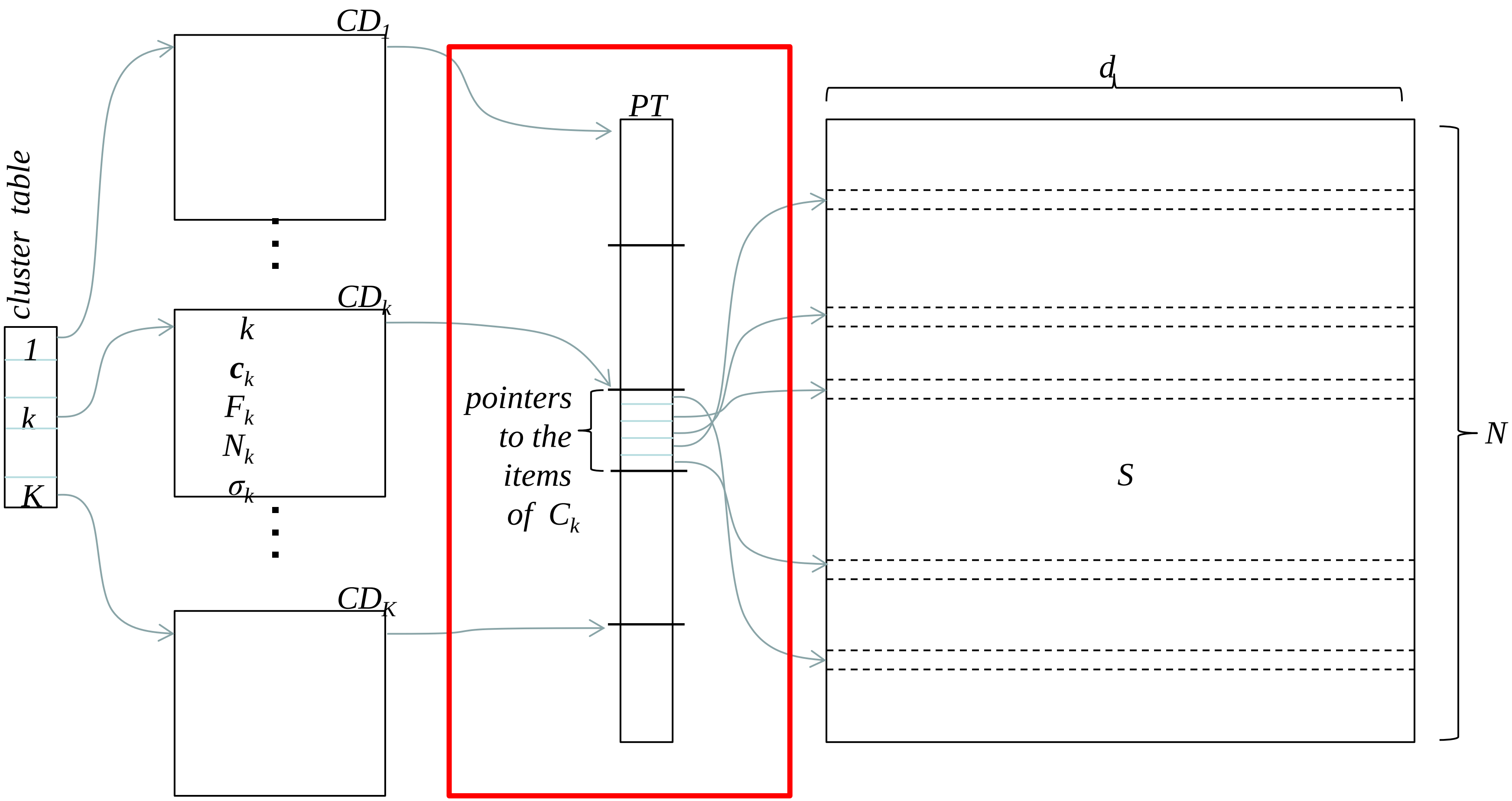
<u>until</u> (RMSSTD starts to grow or it is smaller than a given threshold)

implementation issues: data structure



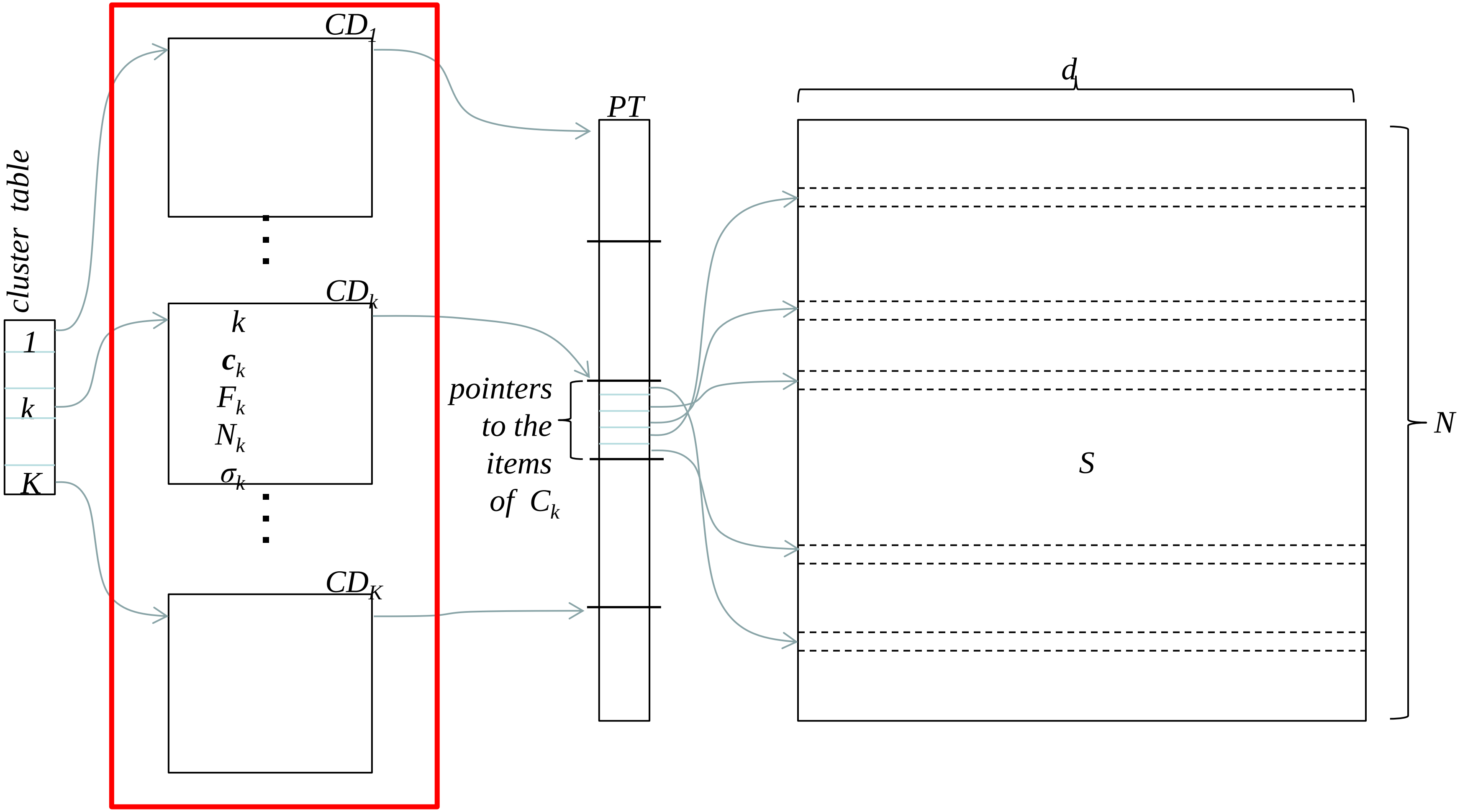
- All the elements $s_n \in S$ are stored, row by row, in a $N \times d$ array.
- In order to improve the computational cost, our method does not change the order of the rows of the array, when the elements must be moved from a cluster to another one

implementation issues: data structure



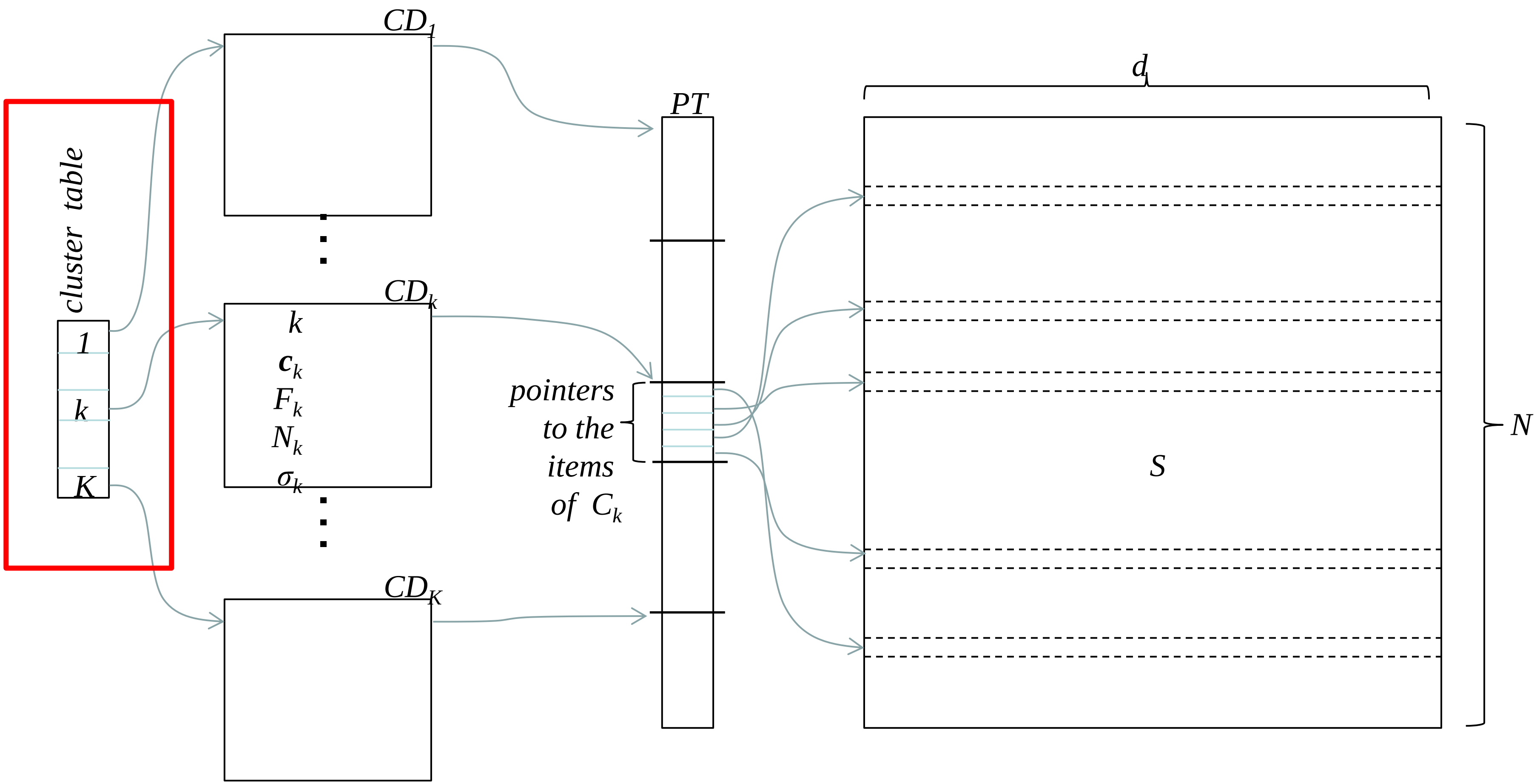
- the composition of each cluster is defined by means of contiguous items in a array PT, pointing to the rows of S representing the elements of the cluster
- All the displacements of elements among clusters are implemented by exchanging only the pointers in the array PT.

implementation issues: data structure



- In order to identify the contiguous items of the array PT pointing to a given cluster C_k, a suitable data structure is defined: a Cluster Descriptor (CD_k) that contains the key features of the cluster

implementation issues: data structure



- the access to the Cluster Descriptors is provided by a Cluster Table (CT), that is a pointers array whose k-th element refers to the cluster descriptor CD_k of the cluster C_k .

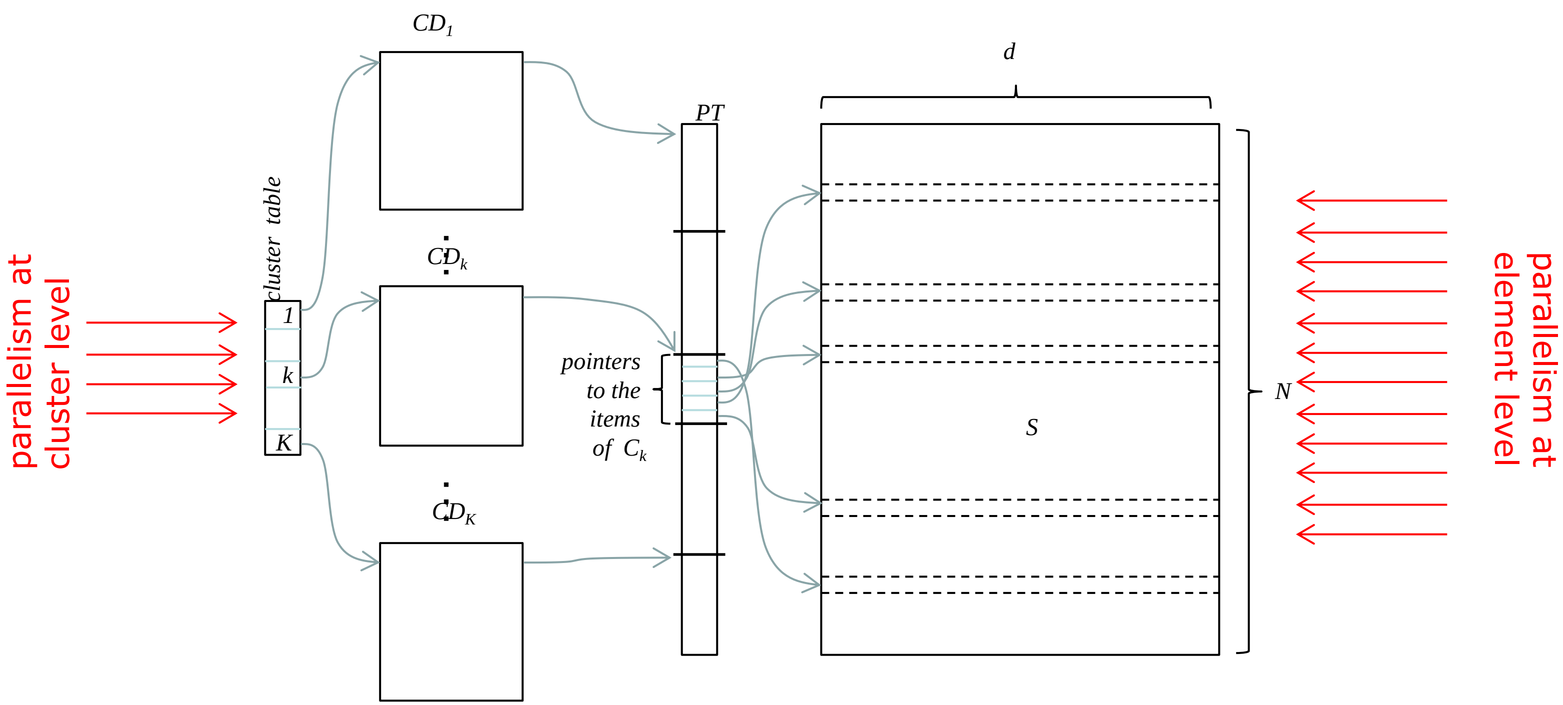
implementation issues: parallelism

In this work we concentrate the attention on multi-core CPUs (shared memory model)

we identified two parallelism levels:

Cluster level. the degree of parallelism if given by the **number of clusters** K, so that it is possible to **distribute the clusters** C_k **among the P threads.**

Element level. the degree of parallelism is given by the **number of elements** N, so that it is possible to **distribute the elements** s_n **among the P threads.**



implementation issues: parallelism

2.4) repeat

2.4.1) Compute the center c_k of each clusters C_k

2.4.2) For each $s_n \in S$ find the cluster $C_{\bar{k}}$ minimizing the Euclidean distance $||s_n - c_k|| \quad k = 1,..,K$

2.4.3) Reassign the elements s_n to the new clusters

until (no change in the reassignment)

kernel of the algorithm

We used the cluster level parallelism in step 2.4.1

We used the element level parallelism in step 2.4.2

Step 2.4.3 is a sequential task (in order to avoid race condition on the array PT)

test on a small problem

Iris data set from the UCI (Univ. California Irvine) Machine Learning Repository

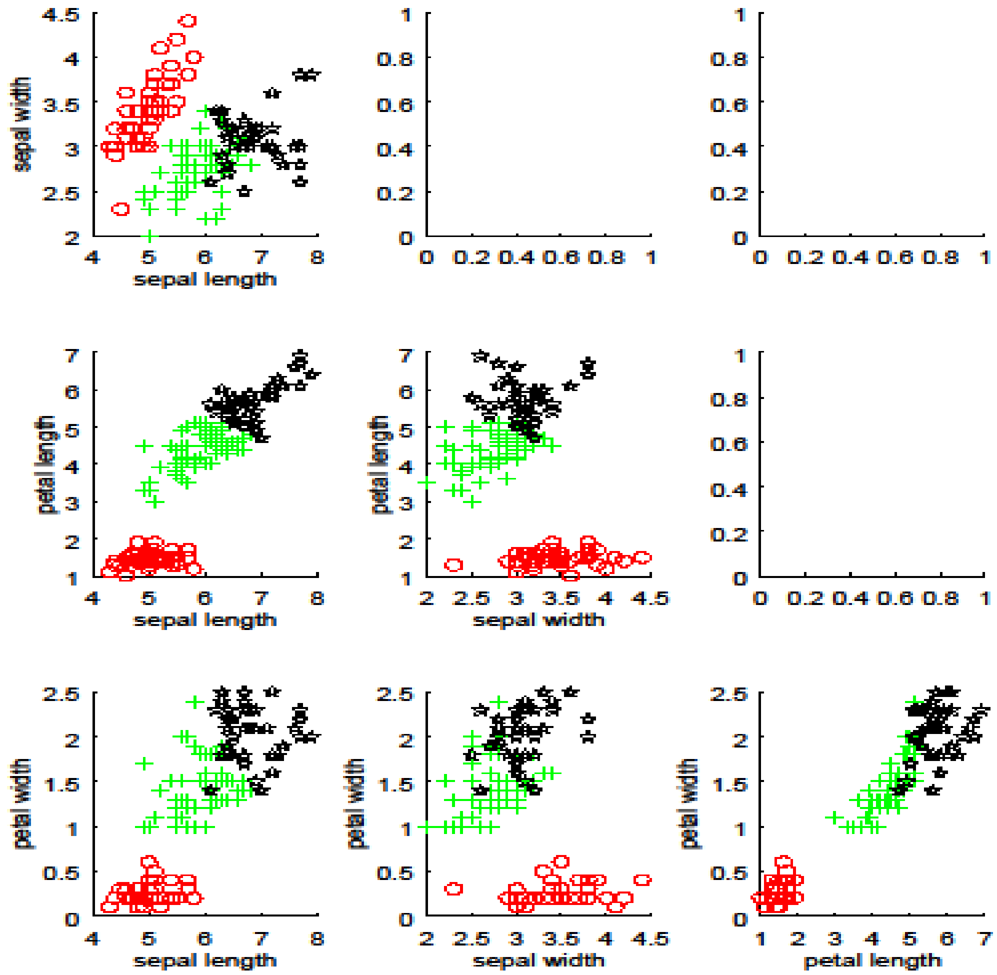
- $N = 150$ instances of iris flowers, divided into $K = 3$ classes of the same dimension $N_k = 50$ elements.
- The items are described on the basis of $d = 4$ attributes: petal's and sepal's width and length.
- Our experiments are aimed to measure the ability of Algorithm 2 to separate the items in three distinct sets and to compare the results with those obtained from Algorithm 1.

	Algorithm 1		**Algorithm 2**	
	N_K	σ_K	N_K	σ_K
C_1	50	0.26	50	0.26
C_2	61	0.30	61	0.30
C_3	39	0.34	39	0.34

same clustering !

Number of items and standard deviation for the three clusters

test on a small problem



each picture refers to a couple of the 4 attributes

test on a large problem

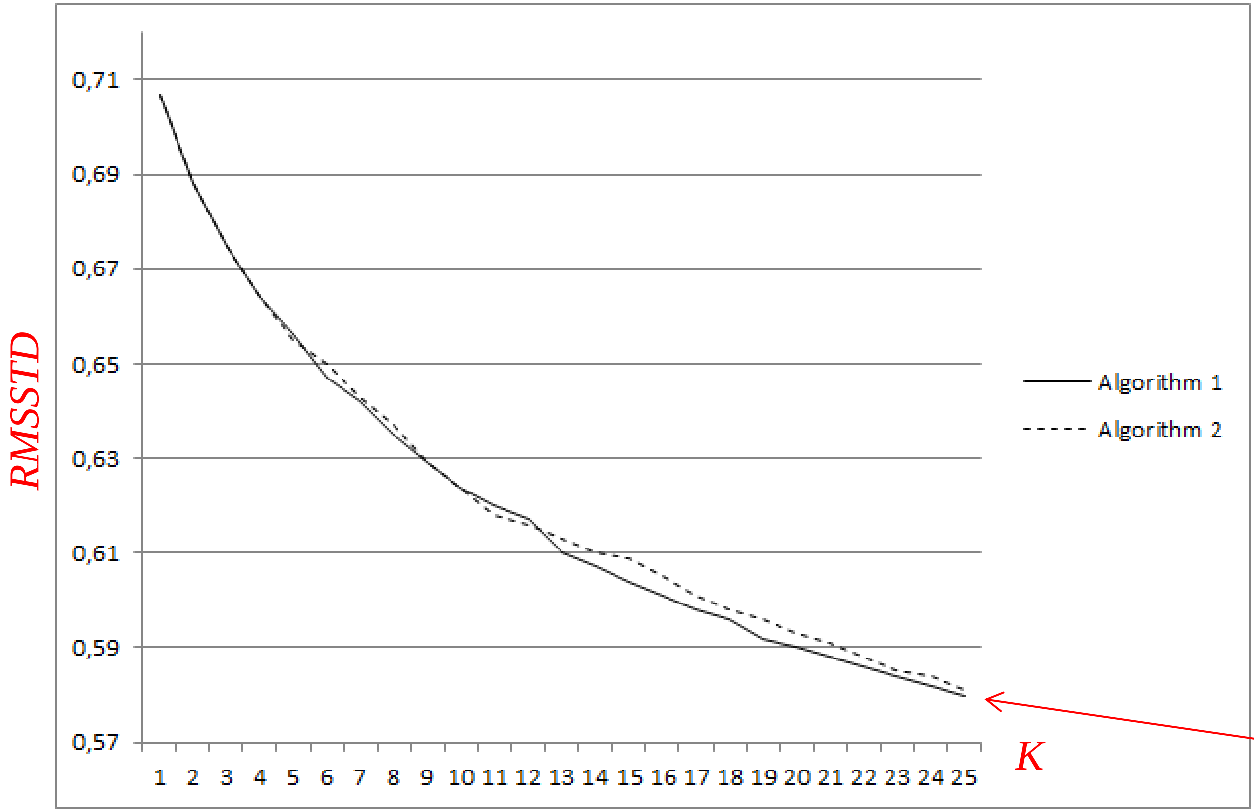
Letter Recognition data set from the UCI (Univ. California Irvine) Machine Learning Repository

- *N = 20000* unique items, each of them representing the black and withe image of an uppercase letter of the English alphabet.
- The character images are based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce an item of the data set.
- Each item was converted into *d = 16* numerical attributes (statistical moments, edge counts,…)

T,2,8,3,5,1,8,13,0,6,6,10,8,0,8,0,8
I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10
D,4,11,6,8,6,10,6,2,6,10,3,7,3,7,3,9
N,7,11,6,6,3,5,9,4,6,4,4,10,6,10,2,8
G,2,1,3,1,1,8,6,6,6,6,5,9,1,7,5,10
S,4,11,5,8,3,8,8,6,9,5,6,6,0,8,9,7
B,4,2,5,4,4,8,7,6,6,7,6,6,2,8,7,10
A,1,1,3,2,1,8,2,2,2,8,2,8,1,6,2,7
J,2,2,4,4,2,10,6,2,6,12,4,8,1,6,1,7
M,11,15,13,9,7,13,2,6,2,12,1,9,8,1,1,8
X,3,9,5,7,4,8,7,3,8,5,6,8,2,8,6,7
O,6,13,4,7,4,6,7,6,3,10,7,9,5,9,5,8
G,4,9,6,7,6,7,8,6,2,6,5,11,4,8,7,8
M,6,9,8,6,9,7,8,6,5,7,5,8,8,9,8,6
R,5,9,5,7,6,6,11,7,3,7,3,9,2,7,5,11
F,6,9,5,4,3,10,6,3,5,10,5,7,3,9,6,9
O,3,4,4,3,2,8,7,7,5,7,6,8,2,8,3,8

accuracy test: Algorithm 1 vs Algorithm 2

Values of the RMSSTD for Algorithm 1 and Algorithm 2 (*K=26* clusters)



less than 1% difference

in about half of the execution time

Time and number of items displaced

Algorithm 1		Algorithm 2	
Disp	*Time (sec)*	*Disp*	*Time (sec)*
1001349	49.3	130801	21.1

efficiency test

- CPU 16-core Intel E7-4850V4 CPU @ 2.1 Ghz
- 16 Gbytes of main memory
- C language, Linux OS, Posix thread library

	Algorithm 1			Algorithm 2		
P	*Time (sec)*	S_P	E_P	*Time (sec)*	S_P	E_P
4	25.94	1.9	0.48	8.11	2.6	0.65
8	15.40	3.2	0.40	4.58	4.6	0.58
12	11.73	4.2	0.35	3.24	6.5	0.54
16	9.66	5.1	0.32	2.57	8.2	0.51

Remember: step 2.4.3 (Reassignment of the elements s_n to the new clusters) is a sequential step, and it is much less expensive in Algorithm 2

conclusions and future works

CONCLUSIONS

- we introduced a parallel adaptive approach to improve the performance of dynamic data clustering with the K-means algorithm.
- Our approach avoids the displacement of similar items already grouped into compact clusters, characterized by small values of the standard deviation.
- The achieved results are very promising, with a clusters quality similar to traditional approaches, with a much lower computational cost and a higher efficiency

FUTURE WORKS

- implementations with other parallel programming models (GPUs, Distributed memories environments)
- applications to real life cases