Designing a Synchronization-reducing Clustering Method on Manycores:

Some Issues and Improvements

Weijian Zheng (wz26@iupui.edu)

November 13, 2017

The Machine Learning in HPC Workshop at SC'17, Denver, CO

IUPUI

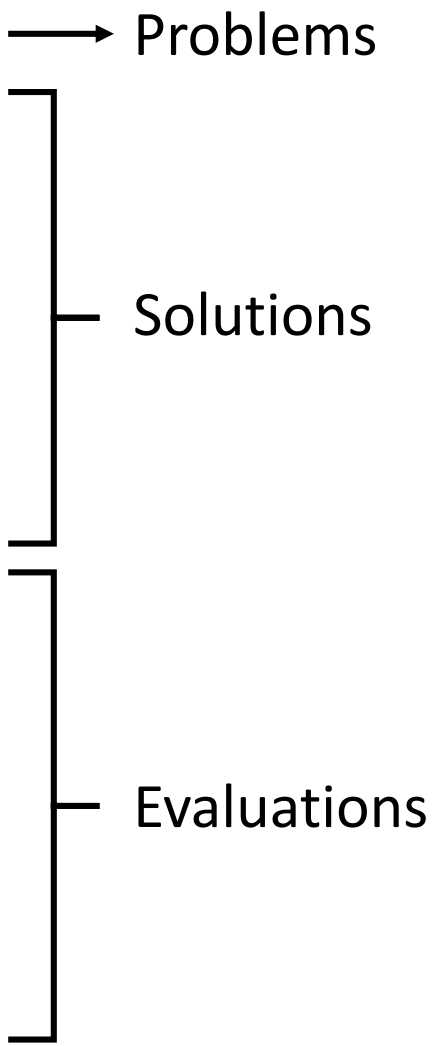
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

SCHOOL OF SCIENCE

A Purdue University School
Indianapolis

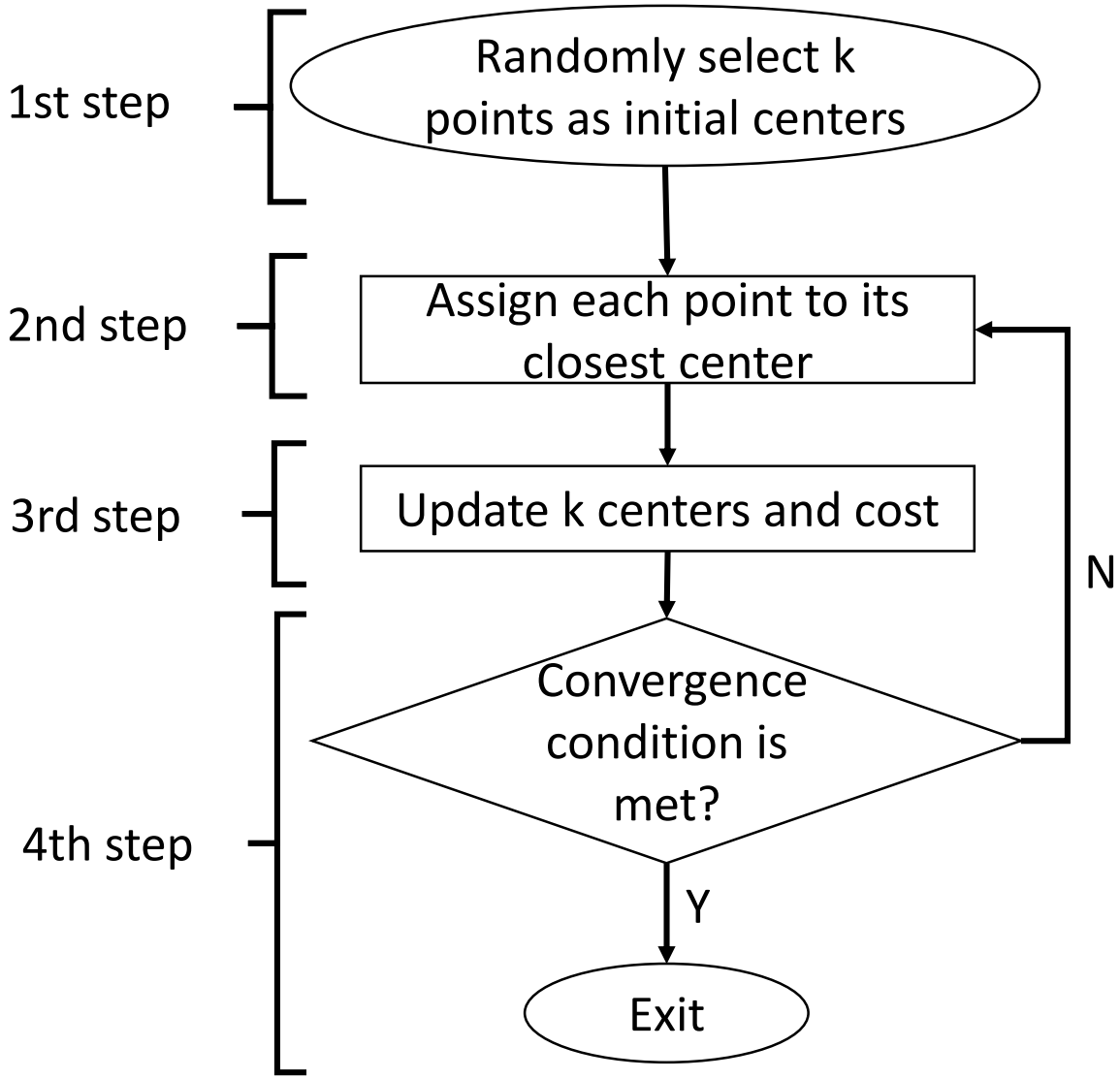
Outline

- Introduction
- Algorithm
  - The synchronization-reducing algorithm
  - The annealing-enhanced algorithm
    - Valley-searching heuristics
- Parallel implementation
- Experimental results
  - Effect of parameters
  - Sequential program performance
  - Parallel performance
- Conclusions & future work



Introduction (1/2)

1. Clustering is one of the most important data summarization methods.
2. Standard K-means works in 4 steps as shown in right figure.



Introduction (2/2)

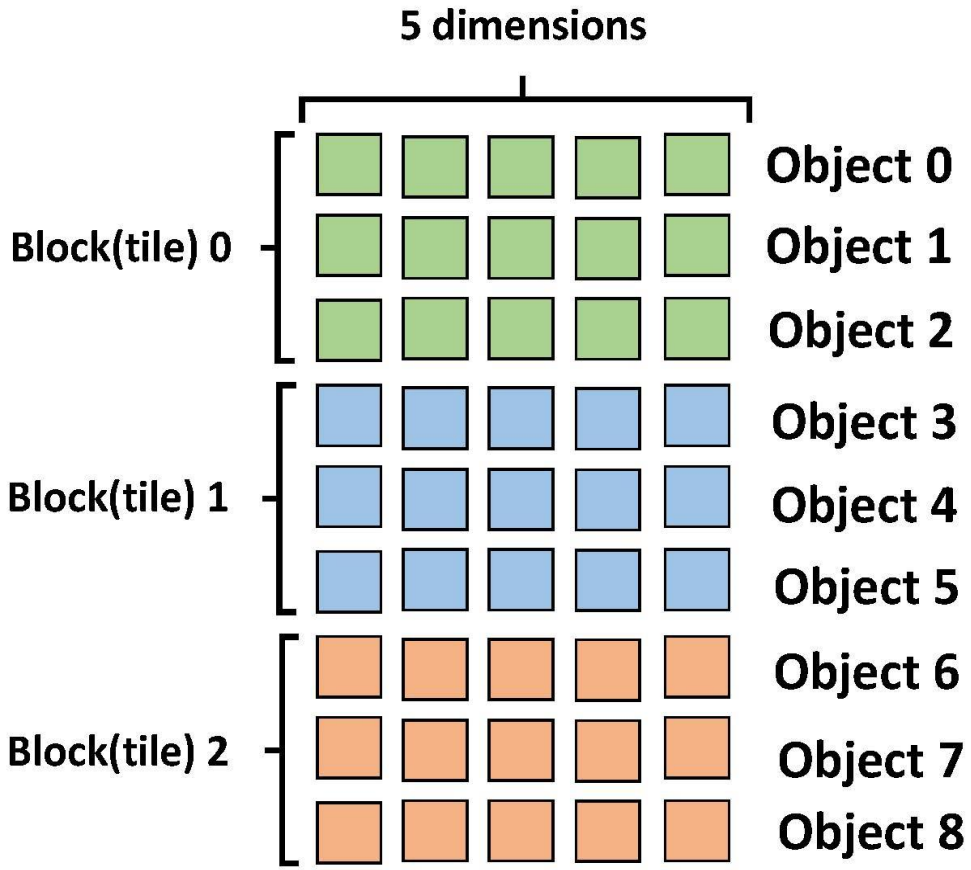
3. Challenges for designing a scalable clustering algorithm:
   A. Frequent synchronization can affect performance.
      - Synchronization is needed after each iteration for parallel K-means.
   B. Visiting all data points at every iteration results in a poor data locality.

4. To solve the previous two challenges, we use:
   A. software blocking (tiled data layout in next slide)
      - solve 2nd challenge
   B. design a new synchronization-reducing clustering algorithm.
      - solve 1st challenge

Data layout

An example of tiled data layout:

- Given 9 objects, each object has 5 dimensions.
- As shown in the figure, each block (or tile) has 3 objects.
- Totally there are 3 blocks (or tiles).



Data layout for 9 objects with block size equal to 3

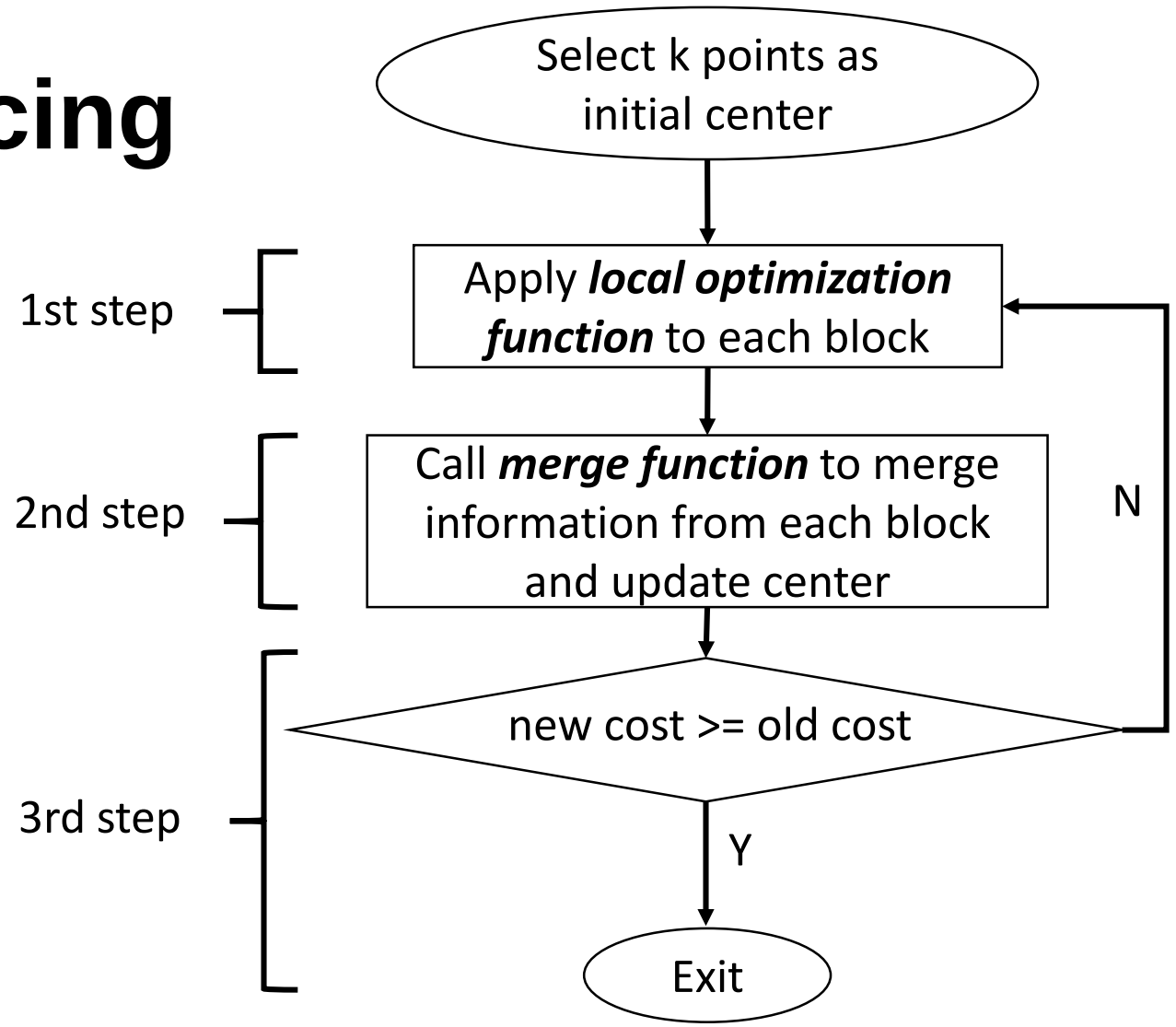
Main body of the synchronization-reducing algorithm

Main body of the algorithm includes 3 steps:

1. **Local optimization function** is used to cluster each block iteratively without any communication

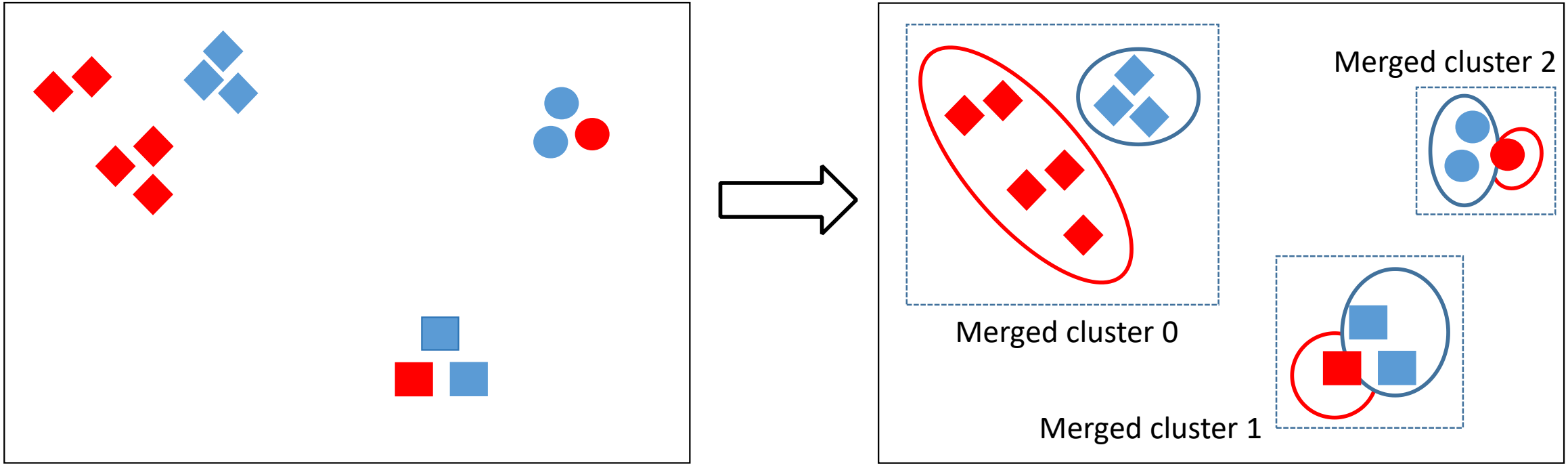
2. **Merge function** is used to merge each block's local information.

3. Converge or not



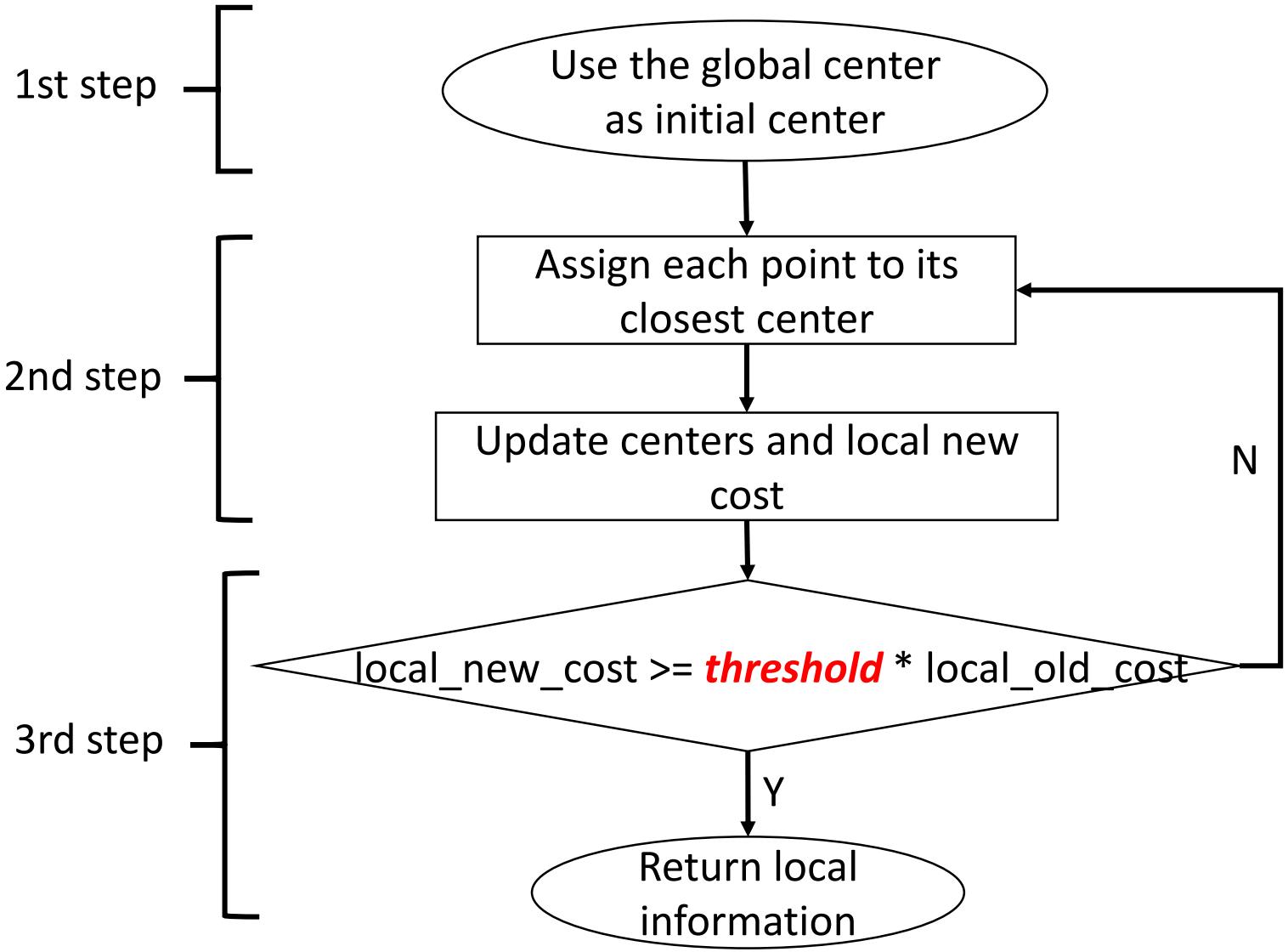
Intuition of the local optimization function

- Two data blocks: red and blue.
- Each red and blue block can find 3 clusters independently.
  - Their merged results are still correct.



Local optimization function

1. There are 3 steps.
2. Threshold is introduced as the stop condition.
   - Most often the progress is too small after a few local iterations



The previous algorithm has some issues…

- For some datasets, it shows a slower convergence rate than K-means.
    - i.e., not always faster
- Hence, we propose the new ***Annealing-enhanced algorithm***

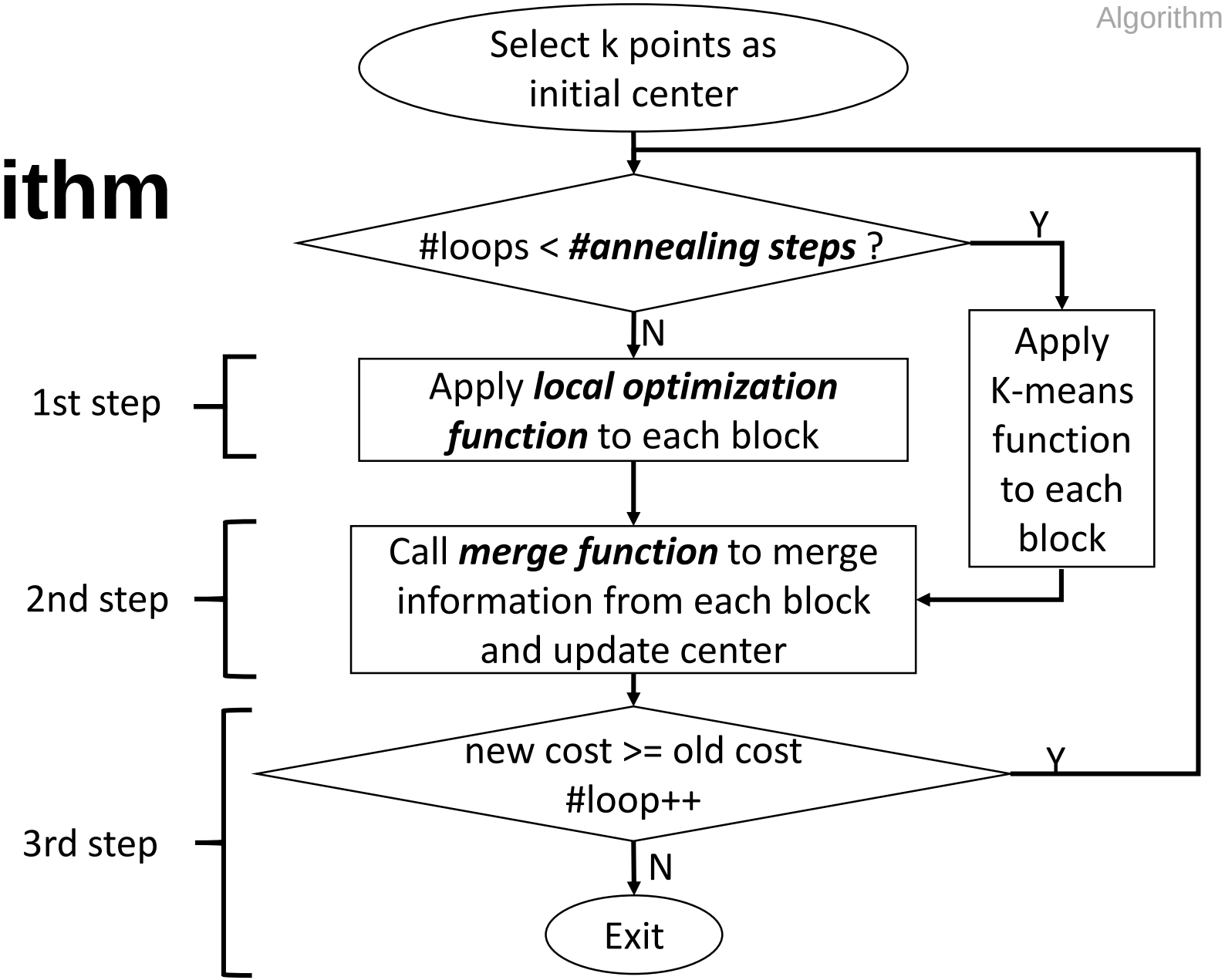
Motivation of using annealing

1. Similar to the game of getting a ball into the lowest crevice.
2. Hard shaking can avoid the ball stuck at local lowest crevice.
3. Also, the ball may roll by itself most of the time.
4. We use frequent synchronization (annealing steps) to simulate “hard shaking”, and use local optimization to simulate “ball rolling on its own”

From https://www.dutchcrafters.com/American-Made-Wooden-Marble-Pyramid/p/54435

The annealing-enhanced algorithm

1. Two types of functions will be used:
   - The previous local optimization function
   - The one-step K-means function on each block
2. 3 steps are included in this algorithm.

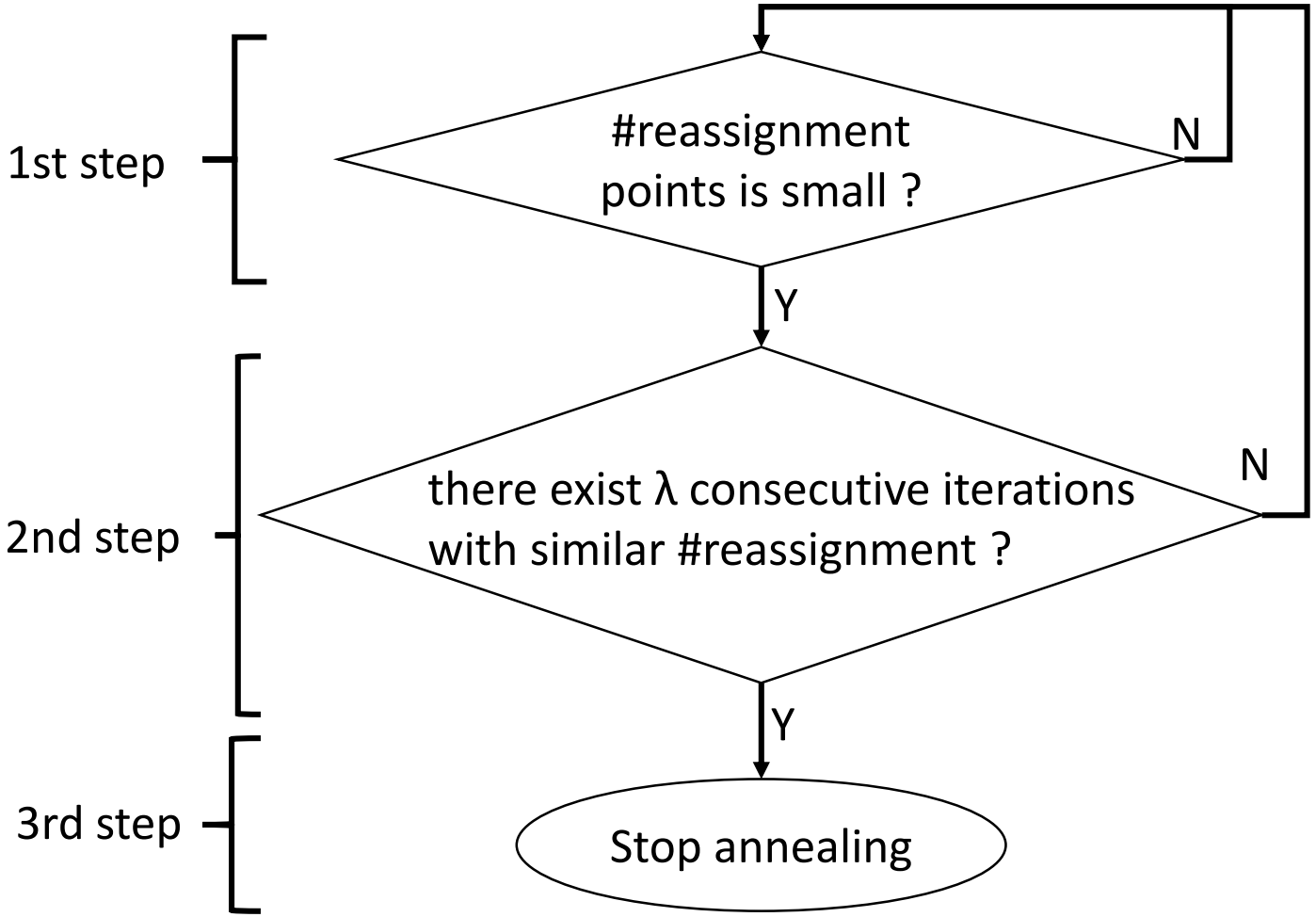


But, how to choose #annealing steps?

- Problem of annealing-enhanced algorithm: Too small or too large number of annealing steps will cause slower convergence.
- How to determine the number of annealing steps?
- We introduce ***Valley-searching heuristics***.

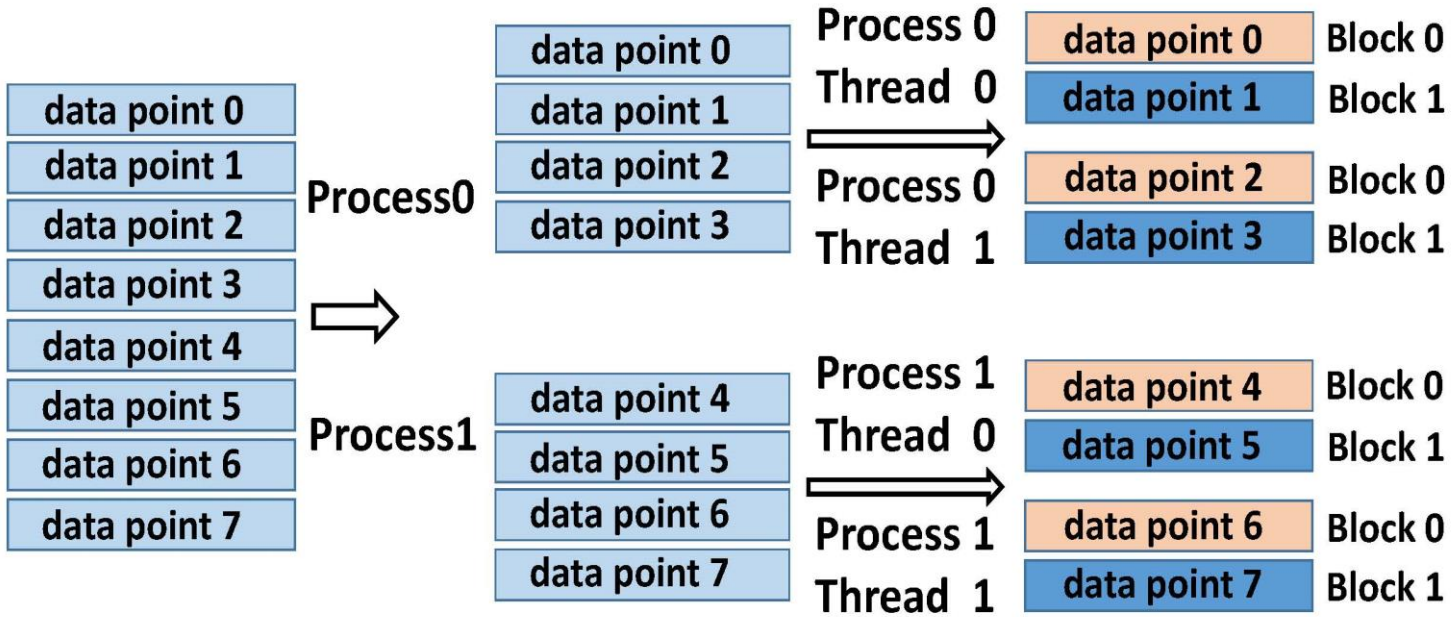
Heuristics to decide the number of annealing steps

1. Using “progression-state” metrics, such as
   1. number of reassignment objects
   2. cost improvement
2. Valley-searching heuristics follows 3 steps:
   1. Find the possible stop places
   2. Verify for the possible stop place
   3. Stop annealing



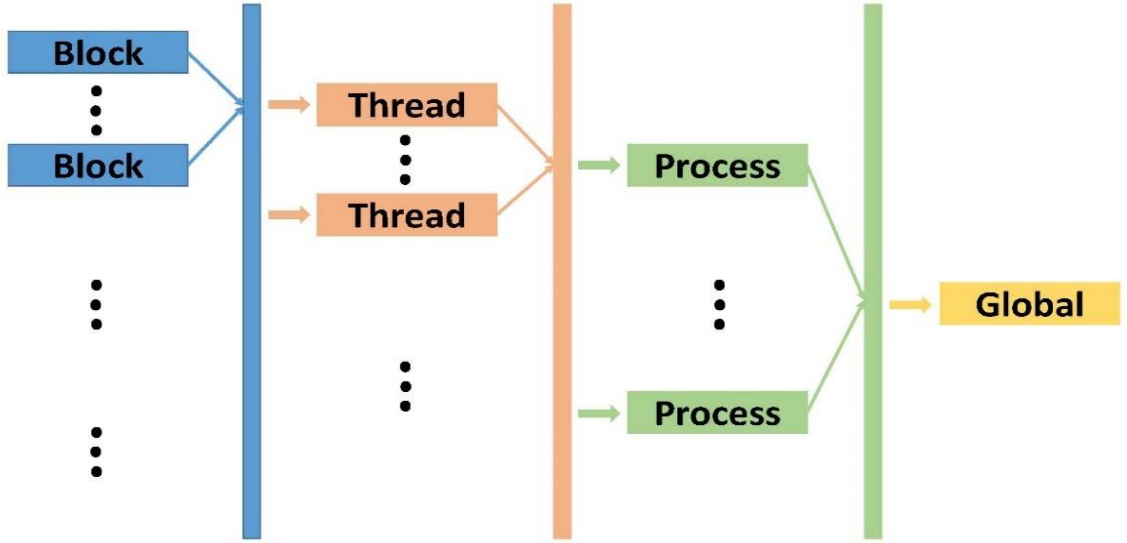
Parallel implementation (1/2)

- Hybrid MPI/Pthread computing model
- Root process is responsible for the distribution of data.



Data distribution example: given 8 data points and each block contains 1 data point, distribution of data into 2 processes (each process has 2 threads), each thread gets 2 blocks

Parallel implementation (2/2)



- 3 levels of merge includes from block to thread, from thread to process, from process to global.

Experimental results

We perform 3 types of experiments using 3 datasets on a Cray XE6/XK7 system.

1. Parameter effect
2. Sequential program performance
3. Parallel performance

Datasets	MNIST	CIFAR10	CIFAR100
#Clusters	10	10	100
#Data points	10,000	60,000	60,000
Dimensions	784	3,072	3,072
Dataset size	17MB	626MB	629MB

Datasets used to test

Parameter effect: #annealing steps

Annealing steps	SSE (sum of square error) cost	#Iterations	Time (seconds)
50	39.884×10^9	140	613.72
75	39.885×10^9	151	502.88
100	39.886×10^9	124	238.8
125	39.884×10^9	208	540.33

- Effect of the number of annealing steps using CIFAR100 dataset
- There exist an optimal number of annealing steps
- 100 is the best number of annealing steps for this case

Parameter effect: stop condition

Threshold (local stop condition)	SSE cost	#Iterations	Time (seconds)
90%	25.332×10^9	32	12.59
95%	25.332×10^9	32	12.59
99%	25.335×10^9	26	10.22
100%	25.541×10^9	21	14.33

- Effect of the stop condition using MNIST dataset
- Large threshold can decrease the #iterations.
- Smallest execution time is obtained by setting threshold=99% in this case.

Sequential program performance

Dataset	K-means		Annealing-enhanced	
	SSE cost	#iterations	SSE cost	#iterations
MNIST	25.322×10^9	106	25.329×10^9	54
CIFAR10	47.437×10^9	88	47.437×10^9	80
CIFAR100	39.884×10^9	201	39.886×10^9	124

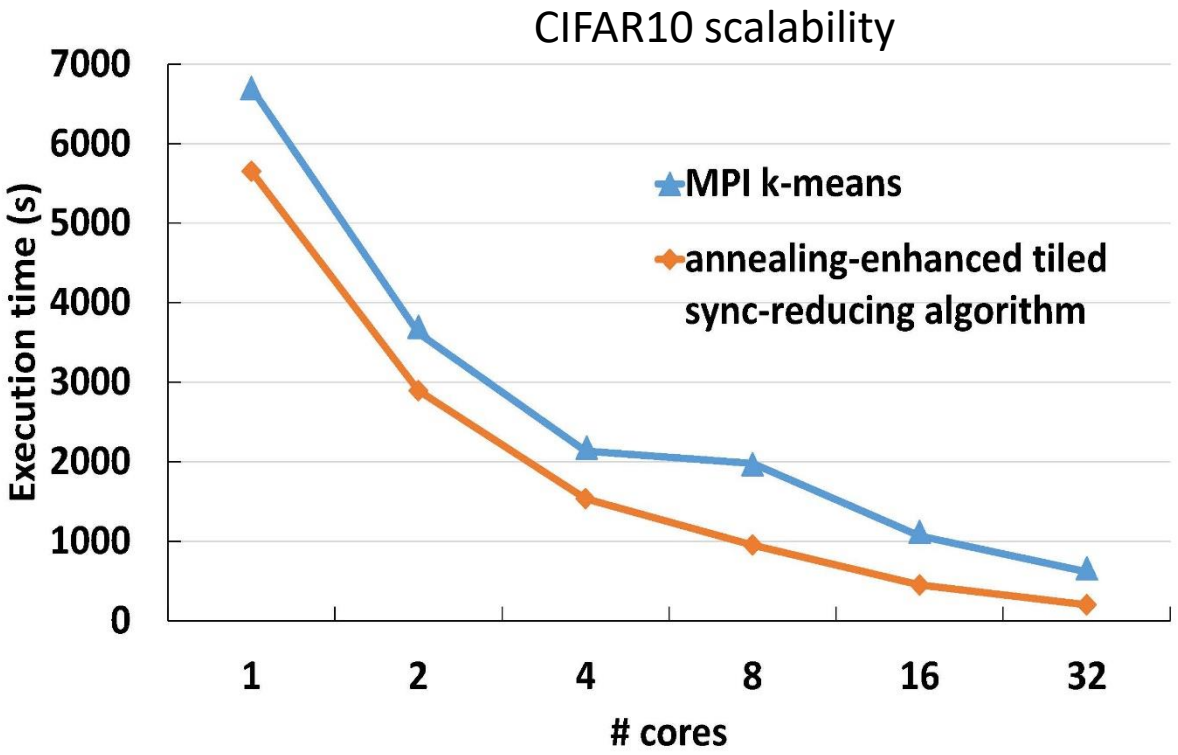
- Cost and #iterations comparison between our algorithm and K-means algorithm
- Our algorithm can obtain similar SSE with less #iterations

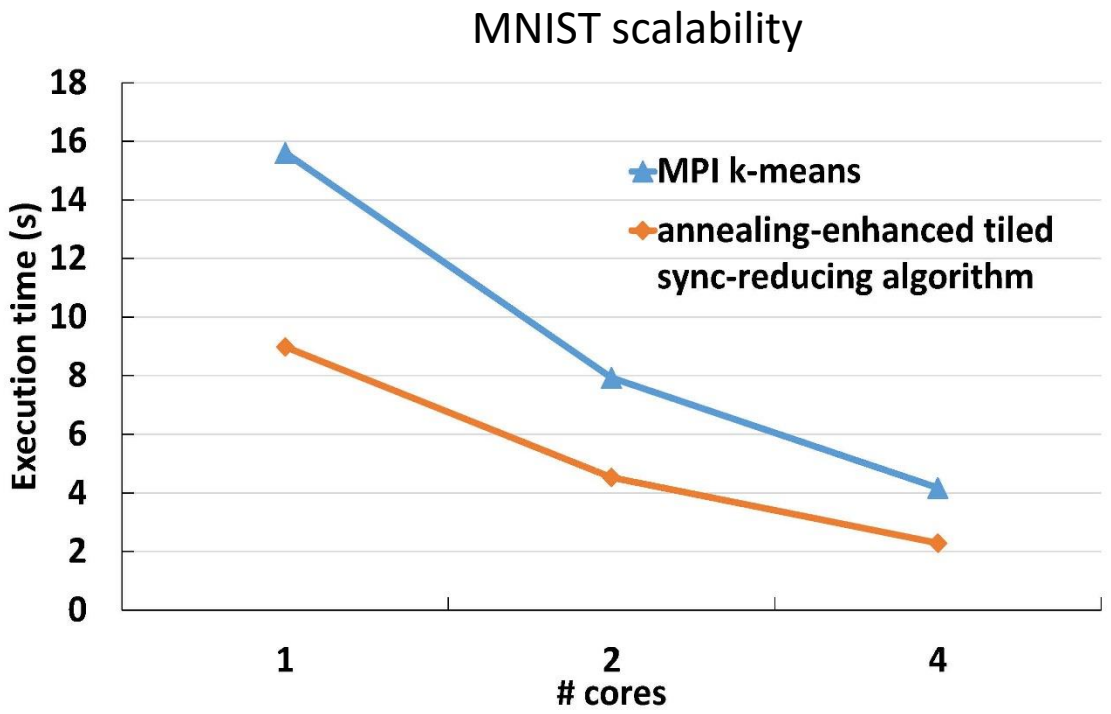
Sequential program performance

Dataset	K-means execution time (seconds)	Annealing-enhanced (seconds)
MNIST	15.6	8.9
CIFAR10	303.5	295
CIFAR100	6700.6	5650

- Sequential version's executing time on different datasets using standard K-means and annealing-enhanced algorithm
- Our annealing-enhanced algorithm is faster than standard k-means

Parallel performance





- Scalability test of CIFAR10 and MNIST dataset using 1 to 32 cores.
- Our implementation presents good scalability and is 19% faster than standard K-means algorithm

Conclusion and future work

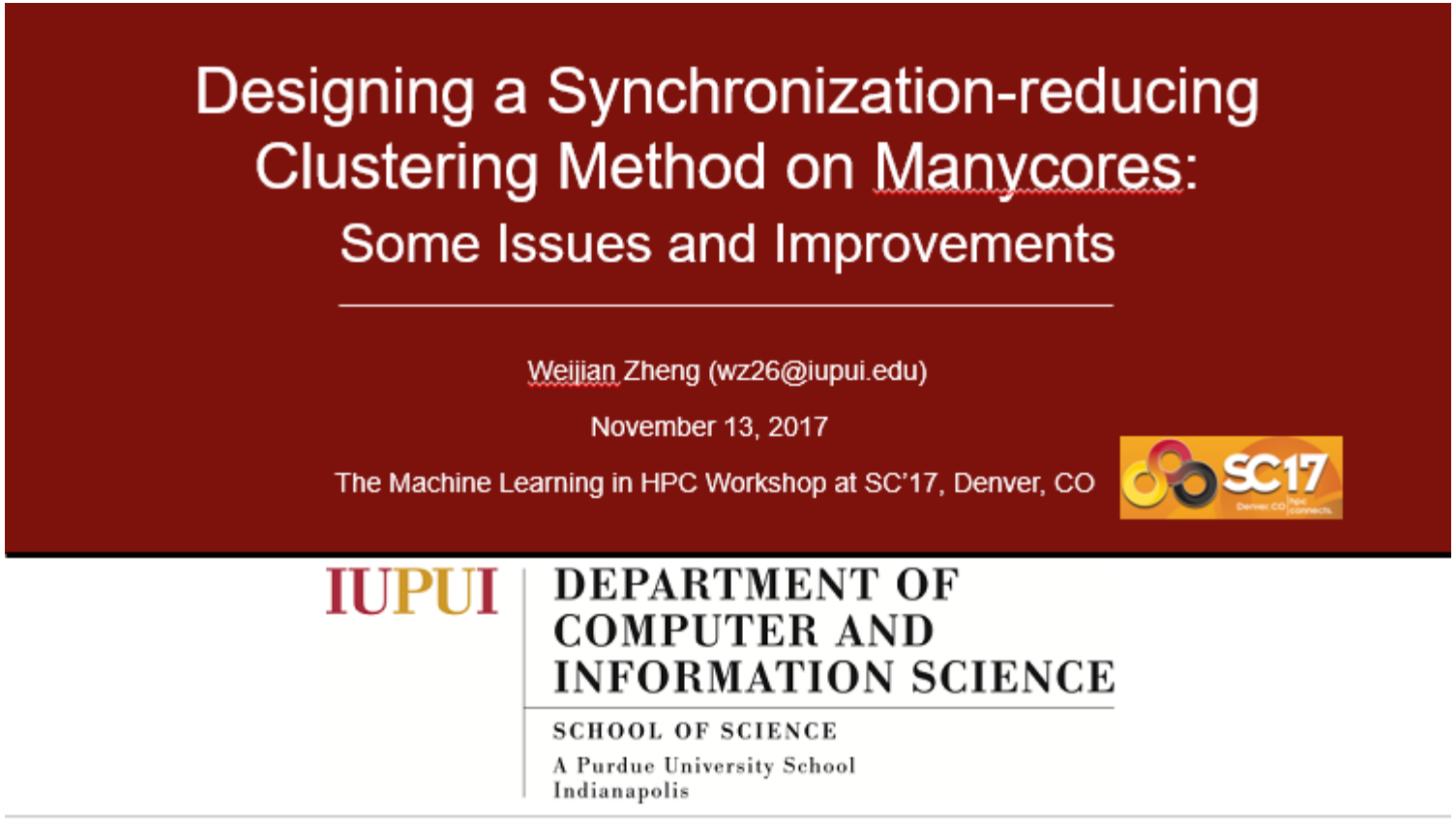
1. To design a more scalable clustering algorithm, we propose a tile-based synchronization-reducing algorithm.

2. Annealing-enhanced algorithm was designed to tackle the problem of original sync-reducing method's slow convergence rate. New heuristics are introduced to support the algorithm.

3. We design and develop a parallel implementation of the annealing-enhanced algorithm using hybrid MPI/Pthread model, providing faster performance than K-means.

4. Our future work is to extend this idea to support other machine learning methods

Thank You!


Designing a Synchronization-reducing Clustering Method on Manycores:
Some Issues and Improvements
Weijian Zheng (wz26@iupui.edu)
November 13, 2017
The Machine Learning in HPC Workshop at SC'17, Denver, CO
SC17
IUPUI
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE
SCHOOL OF SCIENCE
A Purdue University School
Indianapolis