MEMOCODE 2016 Design Contest: k-Means Clustering

Peter Milder
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794–2350
peter.milder@stonybrook.edu

***Abstract*—K-means is a clustering algorithm that aims to group data into k similar clusters. The objective of the 2016 MEMOCODE Design Contest is to implement a system to efficiently partition a large set of multidimensional data using k-means. Contestants were given one month to develop a system to perform this operation, aiming to maximize performance or cost-adjusted performance. Teams were encouraged to consider a variety of computational targets including CPUs, FPGAs, and GPGPUs. The winning team, which was invited to contribute a paper describing their techniques, combined careful algorithmic and implementation optimizations using CPUs and GPUs.**

I. Introduction

This year marks the tenth iteration of the annual MEMOCODE design contest. Since 2007, contests have posed a variety of problems to teams from around the world, focusing on hardware/software solutions. Previous problems have included bioinformatics [1], simulation of NoCs [2], computer vision [3], data mining [4], [5], packet inspection [6], rectangular-to-polar interpolation [7], sorting of encrypted data [8], and matrix-matrix multiplication [9]. This year's problem is to perform k-means clustering on a large data set.

II. Overview

Figure 1 illustrates a simple example performing k-means clustering on two-dimensional data. The black and blue squares represent the data we want to cluster, and the large red circles show the centers of our $k = 2$ clusters. Initially (Step 1), the cluster centers are randomly-generated; the squares are colored black or blue to indicate which of the two clusters each point belongs to. The black-colored points are those closest to the top-left center, and the blue-colored points are those closest to the bottom-right center.

We can easily see in Step 1 that our randomly-chosen centers are not well-placed with respect to the data. The algorithm proceeds by re-calculating the location of each center as the mean value of all the points within its cluster. That is, the top-left red circle's location gets moved to the mean of the locations of all the black squares, and the bottom-right red circle's location gets moved to the mean of the locations of all the blue squares.

The algorithm then proceeds iteratively. In the next iteration (Step 2), the same sequence of operations are performed. First, each of the data points is compared with the locations of the centers, and is assigned to the closest cluster. Then, the centers' locations are again updated by averaging the locations of each element in the cluster.

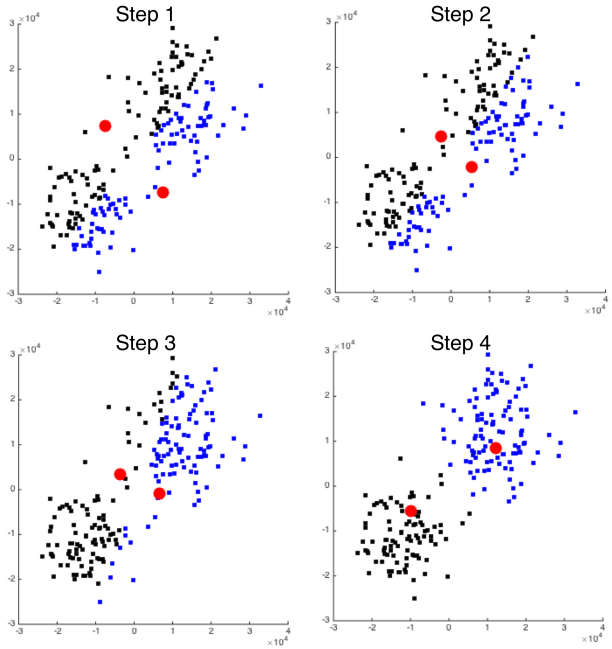

Fig. 1. Example two-dimensional k-means example

As each iteration proceeds, the centers will move to better cluster the data. We can see that by Step 3 the blue and black dots are becoming better separated; by Step 4 they are fully separated. The iterative algorithm will repeat until it converges or a fixed number of iterations is reached.

Problem Description. Let D represent a set of observations consisting of n elements $d_0, \ldots, d_{n-1}$, each representing a point in m-dimensional space. Our goal is to partition these n points into k clusters $S_0, \ldots, S_{k-1}$ in a way that minimizes the distances between each element in D and the center of its nearest cluster.

That is, the goal is to partition D into k clusters S_i that minimizes

$$\sum_{i=0}^{k-1} \sum_{d \in S_i} ||d - \mu_i||^2,$$

where μ_i is the center (average) of all points assigned to cluster S_i, and distance is quantified using squared-Euclidean

distance:

$$||a-b||^2 = \sum_{i=1}^{m}(a_i - b_i)^2, \quad (1)$$

where a_i and b_i represent the i-th dimension of the m-dimensional a and b vectors. Note that we use *squared* Euclidean distance to avoid an unnecessary square-root operation.

Heuristic Algorithm. A common method for performing this clustering is an iterative refinement technique commonly known as Lloyd's algorithm [10]. Given a set of k initial cluster centers $\mu_0, \ldots, \mu_{k-1}$, each iteration performs the following two steps:

1) Assignment: assign each d_j in D to the cluster S_i whose center μ_i is closest to it (based on the squared Euclidean distance).
2) Update: Calculate the new centers μ_i of each cluster S_i as the average location of all points in S_i:

$$\mu_i = \frac{\sum_{d_j \in S_i} d_j}{|S_i|},$$

where $|S_i|$ represents the number of elements assigned to S_i.

These two steps repeat until the algorithm converges or a fixed number of iterations is reached. For this contest, we target a fixed number of iterations of Lloyd's algorithm, or an equivalent distance-metric target if other algorithms are used.

III. Problem Specification

The goal of the contest is to define a system that takes as input: (1) a data set D consisting of n points in m dimensions (where n and m are given), (2) a set of k locations (randomly-chosen) in m-dimensional space which will serve as the initial locations to place the cluster centers μ_i, and (3) the number of iterations to perform if using Lloyd's algorithm (or the target cost metric if using another algorithm).

The system will produce as output: (1) the label for each of the n points (each is an integer between 0 and $k-1$ inclusive), and (2) the final k cluster centers in m-dimensional space.

The timed portion of the solution must start with all input data in main system memory, and it must conclude with all output data in main memory.

Algorithm Choice and Clustering Accuracy. Different algorithms for performing this clustering can result in differing numbers of iterations (i.e., a given algorithm may reach the same quality of results in fewer iterations), or they may converge on a different local minimum. For this reason, we allowed any algorithm to be used so long as it reached a solution "as good" as the naive reference algorithm, as measured by the sum of the squared Euclidean distances of each the point from its nearest center:

$$\text{total distance} = \sum_{i=0}^{k-1} \sum_{d \in S_i} ||d - \mu_i||^2,$$

where μ_i is the center (average) of all points assigned to cluster S_i, and $||a-b||^2$ is defined as in (1).

A target distance computed using this metric was provided to contestants with the reference data sets.

Datatype. Elements in D were given as signed integers. The range of the input values was limited such that every input element can be represented as a 16-bit signed value.

The final output values are: (1) the labels of each input element in D (thus, unsigned integers between 0 and $k-1$), and (2) the set of k cluster centers in m-dimensional space (where each is a signed integer value).

Designers were allowed to choose to optimize the datatype as appropriate. There was no constraint on the order that the results were reported; for convenience the reference implementation code sorts them before storing to a file. (The time for this operation was not included in our timing measurements.)

Functional Reference Implementation. We provided an unoptimized naive software implementation of Lloyd's algorithm to serve as the functional reference for the contestants' optimized implementations. It computes the given number of iterations and stores the result.

This is summarized by the following pseudocode, given k clusters, m dimensions, and T iterations.

```
initialize;
// begin timing here
for t = 1 to T {
   for i = 1 to data set_size {
      minDist = LONG_MAX;
      for j = 1 to k {
         dist =
           calcDist(data[i], cluster[j])
         if (dist<minDist) {
            minDist = dist
            labels[i] = j
         }
      }
      clusterSums[labels[i]] += data[i]
      clusterCounts[labels[i]]++
   }
   for j = 1 to k {
      centers[j] =
        clusterSums[j]/clusterCounts[j]
   }
}
// end timing here
sort and output results;
```

Based on the problem definition, the order of the output values does not matter. For convenience, the functional reference code performs post-processing to sort results based on their location; this was done to allow an easy use of `diff` to determine if two output files are equivalent. This sorting was not required to be in contestants' implementations, and its runtime was ignored.

Data sets. Three data sets were provided to aid in validation and development. Each set contains an m-dimensional input data set D and the initial values of the center locations. Each set is accompanied by a reference solution (computed using Lloyd's algorithm) that provides the sorted center locations and accompanying labels for each input value. Details of

TABLE I. TEST DATA SET CHARACTERISTICS

Name	Size (n)	Dimensions (m)	Clusters (k)	Appx. Input Size	Appx. Output Size
small	500	3	16	3 KiB	1 KiB
medium	100,000	4	128	781 KiB	196 KiB
large	1,048,576	35	256	70 MiB	2 MiB

TABLE II. RUNTIMES FOR THE LARGE DATA SET.

Team	Technology	Platform	Runtime (ms)
IPM	Multicore OpenMP	Intel Xeon E5 2697 v3	2,200
IPM	MPI-2x CPU	Intel Xeon E5 2697 v3	1,375
IPM	Single GPU	NVIDIA GTX 980	312
IPM	Multicore+2x GPU	Xeon E5 and GTX 980	250
IPM	2x GPU	NVIDIA GTX 980	161
IPM	4x GPU	NVIDIA GTX 980	106

the test data are given in Table I. All test data were generated synthetically to have the desired size and characteristics.

IV. THE CONTEST

Participants were given approximately one month to implement a solution using platforms such as FPGAs, GPUs, and CPUs. The solutions were validated using the supplied reference data sets. Performance was measured using the *large* data set. The time taken to initialize data and to sort and read the final result from memory was excluded from the runtime measurement.

The submitted solutions were evaluated using two metrics: pure performance and cost-adjusted performance. The pure-performance metric was based solely on runtime, while the cost-adjusted metric was defined as the product of runtime and system cost. The system cost was determined based on the lowest listed commercial price; if no price were available, the system cost would be estimated by the judges. Contestants were encouraged to include their own estimate of system cost with their submissions.

V. RESULTS

In order to access the reference code and data, contestants were asked to register by e-mail. Five teams consisting of members from five different countries registered; only one team submitted a full working implementation. This submission targeted GPUs and CPUs.

Table II summarizes the submitted results on several platforms. The names and full affiliations of the participants are included at the end of the paper.

As shown in the table, the winning team, from the Institute for Research in Fundamental Sciences (IPM), Iran, implemented their design using Intel Xeon E5 CPUs and NVIDIA GTX 980 GPUs. Their overall best result computed the solution in 106ms using four GPUs. In terms of cost-normalized results, their best solution was the 2x GPU implementation, which was only 1.5x slower than the 4x GPU solution, at half the cost. The IPM team's implementation strategy involved careful parallelization of the problem across available platforms, as well as optimization of the arithmetic required by the problem. Their solution was based on Lloyd's algorithm, and they have contributed an invited paper to these proceedings detailing their techniques [11].

VI. CONCLUSION

The 2016 MEMOCODE Design Contest was to efficiently compute k-means clustering on a large multidimensional data set. The winning team performed effective optimizations involving the algorithmic structure and parallelism.

ACKNOWLEDGMENTS

Thanks to all of the contestants for their hard work, and we congratulate our winners on their excellent results. Thank you also to Yi Deng, Elizabeth Leonard, Klaus Schneider, Sandeep Shukla, and Jean-Pierre Talpin for their help in planning and organizing this contest.

PARTICIPANTS

Thank you to all of the participants. The winning team's members are Saeid Rahmani, Armin Ahmadzadeh, Omid Hajihassani, SeyedPooya Mirhosseini, and Saeid Gorgin, from Institute for Research in Fundamental Sciences (IPM), Iran.

REFERENCES

[1] S. A. Edwards, "MEMOCODE 2012 Hardware/Software codesign contest: DNA sequence aligner," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2012.

[2] D. Chiou, "MEMOCODE 2011 Hardware/Software CoDesign Contest: NoC simulator," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2011.

[3] E. Nurvitadhi, "MEMOCODE 2013 Hardware/Software Co-design Contest: Stereo Matching," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2013.

[4] P. Milder, "MEMOCODE 2014 design contest: k-nearest neighbors with Mahalanobis distance metric," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2014.

[5] ——, "MEMOCODE 2015 design contest: Continuous skyline computation," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2015.

[6] M. Pellauer, A. Agarwal, A. Khan, M. C. Ng, M. Vijayaraghavan, F. Brewer, and J. Emer, "Design Contest Overview: Combined Architecture for Network Stream Categorization and Intrusion Detection (CANSCID)," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2010.

[7] F. Brewer and J. C. Hoe, "2009 MEMOCODE Co-Design Contest," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2009.

[8] P. Schaumont, K. Asanovic, and J. C. Hoe, "MEMOCODE 2008 Co-Design Contest," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2008.

[9] F. Brewer and J. Hoe, "MEMOCODE 2007 co-design contest," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2007.

[10] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. IT-28, no. 2, 1982.

[11] S. Rahmani, A. Ahmadzadeh, O. Hajihassani, S. Mirhosseini, and S. Gorgin, "An efficient multi-core and many-core implementation of k-means clustering," in *ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 2016.