INTRODUCTION
Clustering is a fundamental problem in the analysis of data and distributed algorithms for clustering have been extensively studied as a result of the growth of size in modern datasets as they are too large to fit on one machine. These algorithms usually focus on following aspects:

A. Work with machines having access only to their local dataset.
B. Use small amount of memory and only a few rounds of communication.
C. Have approximation guarantees for the solution they output.

In this work, we focus on distributed algorithms with MapReduce for k-means problem.

Motivation
Reduce and Merge has a key bottleneck:
- In order to have approximation guarantees, machines always need to store k data points.
- All known algorithms require a memory of $kn^\alpha$ if they are to use $O(1/\epsilon)$ rounds of MapReduce computation.
- If a machine sees k points that are all very far from one another, it needs to keep all of them, or else we might lose all the information about one of the clusters, and this could lead to a large objective value.

"Can we partition the data across machines so that different machines work in different regions of space, and thus focus on finding different clusters?"

Goal:
- Smaller space requirement per machine.
- Lesser communication between machines.
- An approximation (bi-criteria) guarantee.

TWO STEP HASHING
Locality Sensitive Hashing
We use the LSH algorithm introduced by Andoni et. al.[3].

- To cover all of $R$, the number of shifts that suffice is $2^O(\sqrt{\log n})$.
- In our setting, we will choose $t = o(\log n / \log \log n)$, and thus the time needed to hash is $n^{o(1)}$.

Product LSH
Given an integer $l$, the product-LSH $PLSH_{l,\epsilon}$ is a hashing scheme that maps a point to a concatenation of $l$ independent copies of $LSH_{\epsilon}$; it thus outputs an $l(t+1)$-tuple of integers. We show the following properties of PLSH.

- Let $\sigma$ be a parameter. Let
  \[ w = 8\alpha(\log n)^{1/2}; t = \log n/(\log \log n)^2; l = 32(\log \log n)^2. \]

Lemma 1. Suppose cluster $C \subseteq U$ has diameter $\leq \sigma$. Then with probability at least 3/4, $PLSH_{l,\epsilon}$ maps all the points in $C$ to the same hash value.

Lemma 2. Let $u, v$ be points that are at distance $\geq 4w/\sqrt{t} = O(\log n \log \log n)$. Then the probability that they have the same hash value is $< 1/n^2$.

Second Step of Hashing
Simply hash each tuple independently and uniformly to an integer in $[lk]$, for a prescribed parameter $l$.

APPROXIMATION ALGORITHM
Algorithm Find-Cover (single machine version)
Input: set of points $U$, rough estimate of optimum $D$.
Output: a subset of points $S$.

for $i = 1 \ldots n$ do
  Hash every point in $U$ into a bin (range $[lk]$) using the two layer hash with parameters $t, \sigma, \epsilon, LK$, where $w = 8\alpha(\log n)^{1/2}$. Let $U'_i$ be the points hashed into bin $i$.
  For each $j$, select a uniformly random subset of $U'_i$ of size $O(1)$ and add them to $S$. (If the number of points in the group is $O(1)$, add all of them to $S$.)
end for

Definition (Adjusted radius). For a cluster $C$ with radius $\rho$, we define the adjusted radius to be $\rho^2 = \rho^2 + \theta + n \theta^2 / |C|$ where $n \theta^2$ is the optimum.

Lemma. Let $C$ be a cluster in the optimal clustering with adjusted radius $\rho'$. With probability $\geq 1/4$, this algorithm outputs a point that is at a distance $\leq \alpha \cdot \rho$ from the center of the cluster $C$, where $\alpha = O(\log \log l)$.

Theorem. Let $S$ be the union of the sets output by $O(\log k)$ independent runs of this algorithm. For $a = O(\log \log l)$, $S$ gives an $(\rho', a(\rho') \log k)$ bi-criteria approximation for $k$-means, w.p. at least 9/10.

DISTRIBUTED IMPLEMENTATION
Theorem. There is a distributed algorithm that runs in $O(\log k + 2)$ rounds of MapReduce, and outputs a bi-criteria approximation with the same guarantee as the single machine algorithm. The number of machines needed is $\text{poly}(\log k, s, \text{min}\{k, s\})$, and the space per machine is $O(s)$. A LOWER BOUND FOR NUMBER OF ROUNDS
Theorem. Let $\alpha$ be any parameter that is $\text{poly}(n)$. Then, any $\alpha$ factor approximation algorithm for $k$-means with $k \geq 2$ that uses $\text{poly}(n)$ machines of memory $s$ requires at least $\log n/\alpha$ rounds of MapReduce.

EXPERIMENTS
Synthetic data: well separated and imbalanced clusters with $10^5$ and $10^6$ points in each dataset. $j$ is number of points sampled from a group in the later case.

Real datasets: Susy dataset and FMA music dataset obtained from UCI repository.

Discussion
- Key limitation - difficult to avoid the polylogarithmic factor in the approximation ratio.
- Assumes an Euclidean setting for the points.
- Future work:
  - obtain a constant factor approximation.
  - extend to other $l_p$ norm based metrics.

REFERENCES